Domain-Independent Dominance of Adaptive Methods

Pedro Savarese
TTI-Chicago
savarese@ttic.edu

David McAllester
TTI-Chicago
mcallester@ttic.edu

Sudarshan Babu
TTI-Chicago
sudarshan@ttic.edu

Michael Maire
University of Chicago
mmaire@uchicago.edu

Abstract

From a simplified analysis of adaptive methods, we derive AvaGrad, a new optimizer which outperforms SGD on vision tasks when its adaptability is properly tuned. We observe that the power of our method is partially explained by a decoupling of learning rate and adaptability, greatly simplifying hyperparameter search. In light of this observation, we demonstrate that, against conventional wisdom, Adam can also outperform SGD on vision tasks, as long as the coupling between its learning rate and adaptability is taken into account. In practice, AvaGrad matches the best results, as measured by generalization accuracy, delivered by any existing optimizer (SGD or adaptive) across image classification (CIFAR, ImageNet) and character-level language modelling (Penn Treebank) tasks. When training GANs, AvaGrad improves upon existing optimizers.[1]

1. Introduction

Deep neural networks are notoriously difficult and costly to train due to the non-convexity of the underlying objective coupled with limitations of first-order methods, like vanishing and shattered gradients [15, 16, 2]. Architectural designs such as normalization layers [19, 1] and residual connections [12] facilitate training by improving gradient statistics, and are broadly used in practice. However, modern architectures often contain modules with different functionalities, such as attention heads [41] and gating mechanisms [17], whose parameter gradients naturally present distinct statistics.

Adaptive gradient methods such as AdaGrad [8] and Adam [21] are particularly suitable for training complex networks, as they designate per-parameter learning rates which are dynamically adapted based on individual gradient statistics collected during training. Although widely adopted, recent works have shown shortcomings in both theoretical and practical aspects of adaptive methods, such as nonconvergence [34] and poor generalization [42].

Moreover, SGD is still dominant when training relatively simple architectures such as ResNets [12, 13] and DenseNets [18], where model-based methods [19, 12] suffice to overcome the obstacles of training deep networks.

A clear trend in the literature is that SGD is more commonly adopted in computer vision tasks, where convolutional networks are prominent [35, 4, 40], while adaptive methods are typically employed in natural language processing tasks, where the most successful networks are either recurrent [17, 30] or attention-based [7, 41]. This situation persists despite the fact that considerable effort has been put into the design of sophisticated adaptive methods, with the goal of providing SGD-like performance in computer vision tasks and formal convergence guarantees.

Newly-proposed optimizers typically offer convergence guarantees for stochastic convex problems [34, 27, 37], but, as we will see, either fail to match SGD's performance when training simpler networks, or behave similarly to SGD and thus underperform Adam when training complex models. The behavior of recently-proposed adaptive methods when training deep networks is unclear due to the scarcity of non-convex guarantees and analyses.

In this paper, we focus on the question of whether an optimizer can be dominant in multiple domains. That is, we are concerned with finding conditions and properties for which an optimization method trains convolutional networks as well as SGD, while at the same time being able to train more complex models as well as Adam. We refer to this property as *domain-independent dominance*.

We start in Section 3 by analyzing the convergence of adaptive methods for stochastic non-convex problems, providing a sufficient condition to guarantee a $\mathcal{O}(1/\sqrt{T})$ convergence rate – the same as SGD. Moreover, we later (Section 8) propose a simple procedure that, given an arbitrary adaptive method, produces a similar optimizer that satisfies said condition and hence offers a guaranteed $\mathcal{O}(1/\sqrt{T})$ convergence rate for stochastic non-convex problems. From this result, we also show how Adam can *provably achieve SGD-like* convergence in stochastic non-convex problems given proper tuning of its adaptability parameter ϵ, and show how this does not contradict [34].

Further inspecting the convergence rate of adaptive meth-

[1] AvaGrad is available at `github.com/lolemacs/avagrad`

ods and the relation with ϵ motivates AvaGrad, a new adaptive method that decouples the learning rate and the adaptability parameter ϵ. In light of our theoretical results, AvaGrad (introduced in Section 4) is uniquely attractive as it virtually removes the interaction between the learning rate – the hyperparameter that requires the most tuning to achieve strong generalization – and the adaptability parameter ϵ – which we show to be strongly connected to convergence.

Section 5 demonstrates through extensive experiments that, against conventional wisdom, Adam can be superior to SGD when training ResNets, even in challenging tasks such as ImageNet [36] classification. The caveat is that achieving SGD-like performance on vision tasks requires extensive tuning of both the learning rate and ϵ, inducing high computational costs due to their interaction.

Our experiments also show that AvaGrad is not merely a theoretical exercise, as it performs as well as both SGD and Adam in their respectively favored usage scenarios without requiring extensive hyperparameter tuning. Section 7 quantifies these differences by measuring suboptimality w.r.t. hyperparameters given a fixed training budget.

Contributions. We offer marked improvements to adaptive optimizers, from theoretical and practical perspectives:

- We show that Adam can *provably converge* for non-convex problems given a proper tuning of its adaptability parameter ϵ. We address the apparent contradiction with [34], providing new insights on the role of ϵ in terms of convergence and performance of adaptive methods.
- Extensive experiments show that Adam can outperform SGD in tasks where adaptive methods have found little success. As suggested by our theoretical results, tuning ϵ is key to achieving optimal results with adaptive methods.
- We propose **AvaGrad**, a theoretically-motivated adaptive method that decouples the learning rate α and the adaptability parameter ϵ. Quantifying the hyperparameter tuning cost using a zeroth-order method, we observe that AvaGrad is significantly cheaper to tune than Adam.

  Matching the generalization accuracy of SGD and other adaptive methods across tasks and domains, AvaGrad offers performance dominance given low tuning budgets.

2. Preliminaries

Notation. For vectors $a = [a_1, a_2, \dots], b = [b_1, b_2, \dots] \in \mathbb{R}^d$ we use $\frac{1}{a} = [\frac{1}{a_1}, \frac{1}{a_2}, \dots]$ for element-wise division, $\sqrt{a} = [\sqrt{a_1}, \sqrt{a_2}, \dots]$ for element-wise square root, and $a \odot b = [a_1 b_1, a_2 b_2, \dots]$ for element-wise multiplication. $\|a\|$ denotes the ℓ_2-norm, while other norms are specified whenever used. The subscript t is used to denote a vector related to the t-th iteration of an algorithm, while i is used for coordinate indexing. When used together, t precedes i: $w_{t,i} \in \mathbb{R}$ denotes the i-th coordinate of $w_t \in \mathbb{R}^d$.

Stochastic Non-Convex Optimization. We consider problems of the form

$$\min_{w \in \mathbb{R}^d} f(w) \coloneqq \mathbb{E}_{s \sim \mathcal{D}} \left[f_s(w) \right] , \tag{1}$$

where $\mathcal{D}$ is a probability distribution over a set $\mathcal{S}$ of "data points", $f_s : \mathbb{R}^d \to \mathbb{R}$ are not necessarily convex and indicate the instant loss for each data point $s \in \mathcal{S}$. As is typically done in non-convex optimization, we assume throughout the paper that f is M-smooth, *i.e.* there exists M such that

$$\|\nabla f(w) - \nabla f(w')\| \leq M \|w - w'\| \tag{2}$$

for all $w, w' \in \mathbb{R}^d$.

We also assume that the instant losses have bounded gradients, *i.e.* $\|\nabla f_s(w)\|_\infty \leq G_\infty$ for some G_∞ and all $s \in \mathcal{S}, w \in \mathbb{R}^d$.

Following the literature on stochastic non-convex optimization [9], we evaluate optimization methods in terms of number of gradient evaluations required to achieve small loss gradients. We assume that the algorithm takes a sequence of data points $S = (s_1, \dots, s_T)$ from which it sequentially and deterministically computes iterates $w_1, \dots, w_T$, using a single gradient evaluation per iterate.

The algorithm then constructs a distribution $\mathcal{P}(t|S)$ over $t \in \{1, \dots, T\}$, samples $t' \sim \mathcal{P}$ and outputs $w_{t'}$. We say an algorithm has a convergence rate of $\mathcal{O}(g(T))$ if

$$\mathbb{E}\left[\|\nabla f(w_t)\|^2 \right] \leq \mathcal{O}(g(T)) , \tag{3}$$

where the expectation is over the draw of the T data points $S \sim \mathcal{D}^T$ and the chosen iterate w_t, $t \sim \mathcal{P}(t|S)$.

Related Work and Adaptive Methods. We consider methods which, at each iteration t, receive or compute a gradient estimate $g_t \coloneqq \nabla f_{s_t}(w_t)$ and perform an update

$$w_{t+1} = w_t - \alpha_t \cdot \eta_t \odot m_t , \tag{4}$$

where $\alpha_t \in \mathbb{R}$ is the **global learning rate**, $\eta_t \in \mathbb{R}^d$ are the **parameter-wise learning rates**, and $m_t \in \mathbb{R}^d$ is the update direction, typically defined in terms of momentum

$$m_t = \beta_{1,t} m_{t-1} + (1 - \beta_{1,t}) g_t \quad \text{and} \quad m_0 = 0 . \tag{5}$$

Note that this definition includes non-momentum methods such as AdaGrad and RMSProp, since setting $\beta_{1,t} = 0$ yields $m_t = g_t$. While in (4) α_t can always be absorbed into η_t, our representation will be convenient throughout the paper. SGD is a special case of (4) when $\eta_t = \vec{1}$, and although it offers no adaptation, it enjoys a convergence rate of $\mathcal{O}(1/\sqrt{T})$ with either constant, increasing, or decreasing learning rates [9]. It is widely used when training relatively simple networks such as feedforward CNNs [12, 18].

Adaptive methods, *e.g.*, RMSProp [6], AdaGrad [8], Adam [21] use $\eta_t = 1/(\sqrt{v_t} + \epsilon)$, with $v_t \in \mathbb{R}^d$ as an exponential moving average of second-order gradient statistics:

$$v_t = \beta_{2,t} v_{t-1} + (1 - \beta_{2,t}) g_t^2 \quad \text{and} \quad v_0 = 0 . \tag{6}$$

Here, m_t and η_t are functions of g_t and can be non-trivially correlated, causing the update direction $\eta_t \odot m_t$ **not** to be an unbiased estimate of the expected update. Precisely this "bias" causes RMSProp and Adam to present nonconvergent behavior even in the stochastic convex setting [34].

We summarize recent advances in adaptive methods as follows, where convergence rates are for stochastic non-convex problems. [46] shows that Adam and newly-proposed Yogi converge as $\mathcal{O}(1/T)$ given a batch size of $\Theta(T)$, a setting that neither captures the small batch sizes used in practice nor fits in the stochastic non-convex optimization framework – their analysis does not yield convergence for a batch size of 1. [5] proves a rate of $\mathcal{O}(\log T/\sqrt{T})$ for AdaGrad and AMSGrad given a decaying learning rate. [27] proposes AdaBound, whose adaptability is decreased during training, but its convergence is only shown for convex problems [37].

The correlation between m_t and η_t is studied in [47], which proposes making the two independent of the sample s_t: the proposed method, AdaShift, is guaranteed to converge in the convex case but at an unknown rate. [43] provides convergence rates for a form of AdaGrad without parameter-wise adaptation, also showing that AdaGrad converges but at an unknown rate. [3] proposes PAdam, which matches or outperforms SGD given proper tuning of a newly-introduced hyparameter – in contrast with their work, we show that even Adam can match SGD given proper tuning and without introducing new hyperparameters.

3. The Role of Adaptivity

We start with a key observation to motivate our studies on how adaptivity affects the behavior of adaptive methods like Adam in both theory and practice: if we let $\alpha_t = \gamma\epsilon$ for some positive scalar γ, then as ϵ goes to ∞ we have

$$\frac{\alpha_t}{\sqrt{v_t}+\epsilon} \to \vec{\gamma}\,, \tag{7}$$

where $\vec{\gamma}$ is the d-dimensional vector with all components equal to γ, and d is the dimensionality of v_t (*i.e.* the total number of parameters in the system). This holds as long as v_t does not explode as $\epsilon \to \infty$, which is guaranteed under the assumption of bounded gradients.

In other words, we have that adaptive methods such as AdaGrad and Adam lose their adaptivity as ϵ increases, and *behave like SGD* in the limit where $\epsilon \to \infty$ *i.e.* all components of the parameter-wise learning rate vector η_t converge to the same value. This observation raises two questions which are central in our work:

1. **How does ϵ affect the convergence behavior of Adam?** It has been shown that Adam does not generally converge even in the linear case [34]. However, as ϵ increases it behaves like SGD, which in turn has well-known convergence guarantees, suggesting that ϵ plays a key, although overlooked, role in the convergence properties of adaptive methods.
2. **Is the preference towards SGD for computer vision tasks purely due to insufficient tuning of ϵ?** SGD is de-facto the most adopted method when training convolutional networks [38, 39, 12, 13, 45, 44], and it is belived that it offers better generalization than adaptive methods [42]. Morever, recently proposed adaptive methods such as AdaBelief [48] and RAdam [25] claim success while underperforming SGD on ImageNet. However, it is not justified to view SGD as naturally better suited for computer vision, because SGD itself can be seen as a special case of Adam.

On the Convergence of Adam, Revisited. We focus on the first question regarding how the convergence behavior of Adam changes with ϵ. As mentioned previously, Reddi *et al.* [34] has shown that Adam can fail to converge in the stochastic convex setting. The next Theorem, stated informally, shows that Adam's nonconvergence also holds in the stochastic non-convex case, when convergence is measured in terms of stationarity instead of suboptimality:

Theorem 1. ***(informal, full version in Appendix B)*** *There exists a stochastic optimization problem (which depends on ϵ) for which Adam does not converge to a stationary point.*

Note that the problem is constructed *adversarially* in terms of ϵ. The problem considered in Theorem 3 of Reddi *et al.* [34], used to show Adam's nonconvergence, has no dependence on ϵ because the proof assumes that $\epsilon = 0$.

The next result, also stated informally, shows that for stochastic non-convex problems *that do not depend on* ϵ, Adam actually converges like SGD as long as ϵ is large enough (or, alternatively, increases during training):

Theorem 2. ***(informal, full version in Appendix E)*** *Adam converges at a $\mathcal{O}(1/\sqrt{T})$ rate for stochastic non-convex problems as long as ϵ is large enough (as a function of the total number of iterations / desired stationarity) or is increased during training (at a rate of $\sqrt{t}$ or faster).*

Together, the two theorems above give a precise characterization of how ϵ affects the theoretical behavior of Adam and other adaptive methods: not only is convergence ensured but a SGD-like rate of $\mathcal{O}(1/\sqrt{T})$ is guaranteed as long as ϵ is large enough. While Adam behaves like SGD in the limit $\epsilon \to \infty$, we show that it suffices for ϵ to be $\mathcal{O}(\sqrt{T})$ to guarantee a SGD-like convergence rate. We believe Theorem 2 is more informative than Theorem 1 for characterizing Adam's behavior, as convergence analyses in the optimization literature typically consider non-adversarial examples.

4. AvaGrad: A New Adaptive Optimizer

We now introduce AvaGrad, a novel adaptive method presented as pseudo-code in Algorithm 1. We describe AvaGrad in this section, but defer its principled motivation to Section 6. Section 5 first presents an experimental study

Algorithm 1 AVAGRAD

```
Input: $w_1 \in \mathbb{R}^d$, $\alpha_t, \epsilon > 0$, $\beta_{1,t}, \beta_{2,t} \in [0, 1)$
1: Set $m_0 = 0, v_0 = 0$
2: for $t = 1$ to $T$ do
3:     Draw $s_t \sim \mathcal{D}$
4:     Compute $g_t = \nabla f_{s_t}(w_t)$
5:     $m_t = \beta_{1,t} m_{t-1} + (1 - \beta_{1,t}) g_t$
6:     $\eta_t = \frac{1}{\sqrt{v_{t-1}} + \epsilon}$
7:     $w_{t+1} = w_t - \alpha_t \cdot \frac{\eta_t}{\left\| \eta_t / \sqrt{d} \right\|_2} \odot m_t$
8:     $v_t = \beta_{2,t} v_{t-1} + (1 - \beta_{2,t}) g_t^2$
9: end for
```

comparing different optimizers, demonstrating AvaGrad's effectiveness across a variety of tasks and domains.

The key difference between AvaGrad and Adam lies in how the parameter-wise learning rates η_t are computed and their influence on the optimization dynamics. In particular, AvaGrad adopts a *normalized* vector of parameter-wise learning rates, which we later show to be advantageous in multiple aspects: it yields better performance and easier hyperparameter tuning in practice, while in theory it results in better convergence rate guarantees.

For convenience, we also account for the dimensionality d of η_t (*i.e.* the total number of parameters in the system) when performing normalization: more specifically, we divide η_t by $\|\eta_t/\sqrt{d}\|_2$ in the update rule, which is motivated by fact that the norm of random vectors increases as $\sqrt{d}$, and also observed to be experimentally robust to changes in d (*e.g.*, networks with different sizes). Alternatively, this normalization can be seen as acting on the global learning rate α_t instead, in which case AvaGrad can be seen as adding an internal, dynamic learning rate schedule for Adam.

Lastly, AvaGrad also differs from Adam in the sense that it updates v_t, the exponential moving average of gradients' second moments, *after* the update step. This implies that parameters are updated according to the second-order estimate of the previous step *i.e.* there is a 1-step *delay* between second-order estimates and parameter updates. Such delay is fully motivated by theoretical analyses, and our preliminary experiments suggest that it does not impact AvaGrad's performance on natural tasks.

5. The Value of Adaptive Gradient Methods

We turn focus to the second question raised in Section 3, on whether tuning ϵ suffices to achieve SGD-like empirical performance regardless of the underlying task and domain.

5.1. Image Classification

We study how adaptive methods perform in computer vision tasks where SGD is the dominant approach – in particular, image classification on the CIFAR [22] and ImageNet [36] datasets, tasks where the state-of-the-art has been consistently surpassed by methods that adopt SGD. Unlike other works in the literature, we perform extensive hyperparameter tuning on ϵ (while also tuning the learning rate α): following our observation that Adam behaves like SGD when ϵ is large, we should expect adaptive methods to perform comparably to SGD if hyperparameter tuning explores large values for ϵ.

For all experiments we consider the following popular adaptive methods: Adam [21], AMSGrad [34], AdaBound [27], AdaShift [47], RAdam [25], AdaBelief [48], and AdamW [26]. We also report results of AvaGrad, our newly-proposed adaptive method, along with its variant with decoupled weight decay [26], which we refer to as AvaGradW.

CIFAR. We train a Wide ResNet-28-4 [45] on the CIFAR dataset [22], which consists of 60,000 RGB images with 32×32 pixels, and comes with a standard train/test split of 50,000 and 10,000 images. Following [45], we normalize images prior to training. We augment the training data with horizontal flips and by sampling 32×32 random crops after applying a 4-pixel padding to the images.

We adopt the same learning rate schedule as [45], decaying α by a factor of 5 at epochs 60, 120 and 160 – each network is trained for a total of 200 epochs on a single GPU. We use a weight decay of 0.0005, a batch size of 128, a momentum of 0.9 for SGD, and $\beta_1 = 0.9, \beta_2 = 0.999$ for each adaptive method.

We select a random subset of 5,000 samples from CIFAR-10 to use as the validation set when tuning α and ϵ of each adaptive method. We perform grid search over a total of 441 hyperparameter settings, given by all combinations of $\epsilon \in \{10^{-8}, 2 \cdot 10^{-8}, 10^{-7}, \dots, 100\}$ and $\alpha \in \{5 \cdot 10^{-7}, 10^{-6}, 5 \cdot 10^{-6}, \dots, 5000\}$.

Results of our hyperparameter tuning procedure agree with our hypothesis: for this specific setting, adaptive methods perform best with aggressive values for ϵ, ranging from 0.1 (Adam, AMSGrad) to 10.0 (AvaGrad, AdamW) – values drastically larger then the default $\epsilon = 10^{-8}$. In terms of the learning rate, Adam and AMSGrad perform best with $\alpha = 0.1$, a value 100 times larger than the default.

Next, we fix the best (α, ϵ) values found for each method and train a Wide ResNet-28-10 on both CIFAR-10 and CIFAR-100, this time evaluating the test performance. We do not re-tune α and ϵ for adaptive methods due to the practical infeasibility of training a Wide ResNet-28-10 roughly 8000 times. We tune the learning rate α of SGD using the same search space as before, and confirm that the learning rate $\alpha = 0.1$ commonly adopted when training ResNets [12, 13, 45] performs best in this setting.

The leftmost columns of Table 1 present results: on CIFAR-10, SGD (3.86%) is outperformed by Adam (3.64%) and AvaGrad (3.80%), while on CIFAR-100 Adam (18.96%), AMSGrad (18.97%), AdaShift (18.88%), AvaGrad (18.76%), and AvaGradW (19.04%) all outperform

SGD (19.05%). These results disprove the conventional wisdom that adaptive methods are not suited for computer vision tasks such as image classification. While tuning ϵ, a step typically overlooked or skipped altogether in practice, suffices for adaptive methods to outperform SGD (and hence can be a confounding factor in comparative studies), our results also suggest that adaptive methods might require large compute budgets for tuning to perform optimally on some tasks.

ImageNet. To further validate that adaptive methods can indeed outperform SGD in settings where they have not been historically successful, we consider the challenging task of training a ResNet-50 [13] on the ImageNet dataset [36]. The task consists of 1000-way classification given 1.2M training and 50,000 validation images. Following [11], we perform scale/color transformations for training and use single 224×224 crops to compute the top-1 validation error.

We transfer the hyperparameters from our CIFAR experiments for all methods. The network is trained for 100 epochs with a batch size of 256, split between 4 GPUs, where the learning rate α is decayed by a factor of 10 at epochs 30, 60 and 90, and we also adopt a weight decay of 10^{-4}. Note that the learning rate of 0.1 adopted for SGD agrees with prior work that established new state-of-the-art results on ImageNet with residual networks [12, 13, 44].

SGD yields 24.01% top-1 validation error, underperforming Adam (23.45%), AMSGrad (23.46%), RAdam (23.60%), AvaGrad (23.58%) and AvaGradW (23.49%), *i.e.* 5 out of the 8 adaptive methods evaluated on this task. We were unable to train with AdaShift due to memory constraints: since it keeps a history of past gradients, our GPUs ran out of memory even with a reduced batch size of 128, meaning that circumventing the issue with gradient accumulation would result in considerably longer training time.

The third column of Table 1 summarizes the results. In contrast to numerous papers that surpassed the state-of-the-art on ImageNet by training networks with SGD [38, 39, 12, 13, 45, 44], our results show that adaptive methods can yield superior results in terms of generalization performance as long as ϵ is appropriately chosen. Most strikingly, Adam outperforms AdaBound, RAdam, and AdaBelief: sophisticated methods whose motivation lies in improving the performance of adaptive methods.

5.2. Language Modelling

We now consider a task where state-of-the-art results are achieved by adaptive methods with small values for ϵ and where SGD has little success: character-level language modelling with LSTMs [17] on the Penn Treebank dataset [29, 31]. We adopt the 3-layer LSTM [17] model from Merity *et al.* [30] with 300 hidden units per LSTM layer.

We first perform hyperparameter tuning over all combinations of $\epsilon \in \{10^{-8}, 5 \cdot 10^{-7}, \dots, 100\}$ and $\alpha \in \{2 \cdot 10^{-4}, 10^{-3}, \dots, 20\}$, training each model for 500 epochs and decaying α by 10 at epochs 300 and 400. Since ϵ affects AdaBelief differently and its official codebase recommends values as low as 10^{-16} for some tasks [2], we adopt a search space where candidate values for ϵ are smaller by a factor of 10^{-8} *i.e.* starting from 10^{-16} instead of 10^{-8}.

We use a batch size of 128, BPTT length of 150, and weight decay of 1.2×10^{-6}. We also employ dropout with the recommended settings for this model [30]. Not surprisingly, our tuning procedure returned small values for ϵ as being superior for adaptive methods, with Adam, AMSGrad, and AvaGrad performing optimally with $\epsilon = 10^{-8}$.

Next, we train the same 3-layer LSTM but with 1000 hidden units, transferring the (α, ϵ) configuration found by our tuning procedure. For SGD, we again confirmed that the transferred learning rate performed best on the validation set when training the wider model.

Results in Table 1 show that only AvaGrad and AvaGradW outperform Adam, achieving test BPCs of 1.179 and 1.175 compared to 1.182. Combined with the previous results, we validate that, depending on the underlying task, adaptive methods might require vastly different values for ϵ to perform optimally, but, given enough tuning, are indeed capable of offering best overall results across domains.

We also observe that AdaBound, RAdam, and AdaBelief all visibly underperform Adam in this setting where adaptivity (small ϵ) is advantageous, even given extensive hyperparameter tuning. RAdam, and more noticeably AdaBound, perform poorly in this task. We hypothesize that this is result of RAdam incorporating learning rate warmup (see Ma & Yarats [28] for more details), which is not typically employed when training LSTMs, and AdaBound's adoption of SGD-like dynamics early in training [37].

5.3. Generative Adversarial Networks

Finally, we consider a task where adaptivity is not only advantageous, but often seen as necessary for successful training: generative modelling with GANs. We train a Geometric GAN [24], *i.e.* a DCGAN model [33] with the hinge loss, on the CIFAR-10 dataset to perform image generation. We do not apply gradient penalties.

We adopt a batch size of 64 and train the networks for a total of 60,000 steps, where the discriminator is updated twice for each generator update. We train the GAN model with the same optimization methods considered previously, performing hyperparameter tuning over $\epsilon \in \{10^{-8}, 10^{-6}, 10^{-4}\}$ and $\alpha \in \{10^{-5}, 2 \cdot 10^{-5}, 10^{-4}, \dots, 0.1\}$ for each adaptive method, and $\alpha \in \{10^{-6}, 2 \cdot 10^{-6}, 10^{-5}, \dots, 1.0\}$ for SGD.

The performance of each model is measured in terms of the Fréchet Inception Distance (FID) [14] computed from a total of 10,000 generated images. Results are summarized in Table 1, showing that AvaGrad offers a significant improvement in terms of FID over all other methods,

[2] `github.com/juntang-zhuang/Adabelief-Optimizer`, ver. 9b8bb0a

Table 1: Test performance of standard models on benchmark tasks, when trained with different optimizers. Gray background indicates the optimization method (baseline) adopted by the paper that proposed the corresponding network model. The best task-wise results are in bold, while other improvements over the baselines are underlined. Numbers in parentheses indicate standard deviation over three runs. Across tasks, AvaGrad closely matches or exceeds the results delivered by existing optimizers, and offers notable improvement in FID when training GANs.

	CIFAR-10 Test Err%	CIFAR-100 Test Err %	ImageNet Val Err %	Penn Treebank Test BPC ↓	Penn Treebank Test BPC ↓	CIFAR-10 FID ↓
Model	WRN 28-10	WRN 28-10	ResNet-50	3xLSTM(300)	3xLSTM(1000)	GGAN
SGD	3.86 (0.08)	19.05 (0.24)	24.01	1.404	1.237 (0.000)	133.0
Adam	**3.64 (0.06)**	18.96 (0.21)	**23.45**	1.377	1.182 (0.000)	43.0
AMSGrad	3.90 (0.17)	18.97 (0.09)	23.46	1.385	1.187 (0.001)	41.3
AdaBound	5.40 (0.24)	22.76 (0.17)	27.99	—	2.891 (0.041)	247.3
AdaShift	4.08 (0.11)	18.88 (0.06)	OOM	1.395	1.199 (0.001)	43.7
RAdam	3.89 (0.09)	19.15 (0.13)	23.60	—	1.349 (0.003)	42.5
AdaBelief	3.98 (0.07)	19.08 (0.09)	24.11	1.377	1.198 (0.000)	44.8
AdamW	4.11 (0.17)	20.13 (0.22)	26.70	1.401	1.227 (0.003)	—
AvaGrad	3.80 (0.02)	**18.76 (0.20)**	23.58	**1.375**	1.179 (0.000)	**35.3**
AvaGradW	3.97 (0.02)	19.04 (0.37)	23.49	**1.375**	**1.175 (0.000)**	—

achieving an improvement of 7.7 FID over Adam (35.3 against 43.0). Note that the performance achieved by Adam matches other sources[3] [20], and Adam performed best with $\alpha = 0.0002, \epsilon = 10^{-6}$ in our experiments, closely matching the commonly-adopted values in the literature.

6. Decoupling α and ϵ with AvaGrad

The results in the previous section establish the importance of optimizing ϵ when using adaptive methods, and how not tuning ϵ can be a confounding factor when comparing different adaptive optimizers.

A key obstacle to proper tuning of ϵ is its interaction with the learning rate α: as discussed in Section 3, 'emulating' SGD with a learning rate γ can be done by setting $\alpha = \gamma\epsilon$ in Adam and then increasing ϵ: once its value is large enough (compared to v_t), scaling up ϵ any further will not affect Adam's behavior as long as α is scaled up by the same multiplicative factor. Conversely, when ϵ is small (compared to components of v_t), we have that $\sqrt{v_t} + \epsilon \approx \sqrt{v_t}$, hence decreasing ϵ even further will not affect the optimization dynamics as long as α remains fixed.

This suggests the existence of two distinct regimes for Adam (and other adaptive methods): an adaptive regime, when ϵ is small and there is no interaction between α and ϵ, and a non-adaptive regime, when ϵ is large and the learning rate α must scale linearly with ϵ to preserve the optimization dynamics. The exact phase transition is governed by v_t *i.e.* the second moments of the gradients, which depends not only on the task but also on the model.

[3] github.com/POSTECH-CVLab/PyTorch-StudioGAN

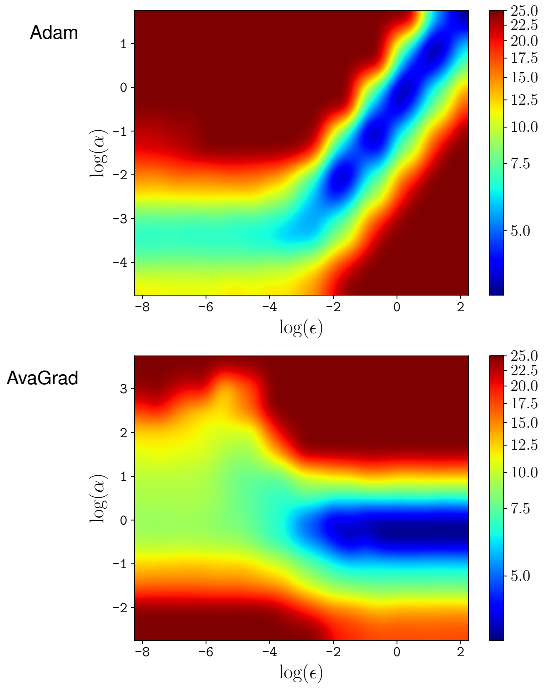

Figure 1: Performance of Adam and AvaGrad with different learning rate α and adaptability parameter ϵ, measured in terms of validation error on CIFAR-10 of Wide ResNet 28-4. Best performance is achieved with low adaptability/large ϵ.

By normalizing the parameter-wise learning rates η_t at each iteration, AvaGrad guarantees that the magnitude of the effective learning rates is independent of ϵ, essentially decoupling it from α. With AvaGrad, α governs optimization

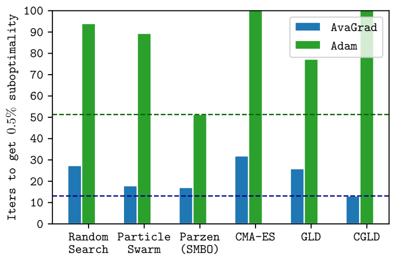
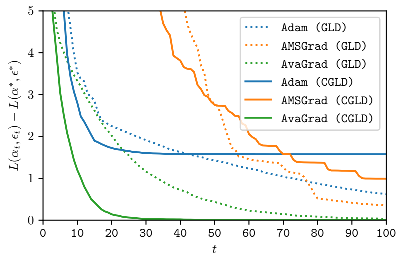
Figure 2: ***Left:*** Iterations to achieve 0.5% suboptimality, measured in terms of validation accuracy on CIFAR-10, for Adam and AvaGrad when tuning α and ϵ with various standard hyperparameter optimizers. ***Right:*** Suboptimality (gap in validation accuracy) when optimizing α and ϵ with GLD/CGLD, as a function of trials (*i.e.* validation accuracy evaluations for a value of (α, ϵ)): AvaGrad is significantly cheaper to tune than Adam, being especially efficient when adopting Coordinate GLD due to its hyperparameter separability.

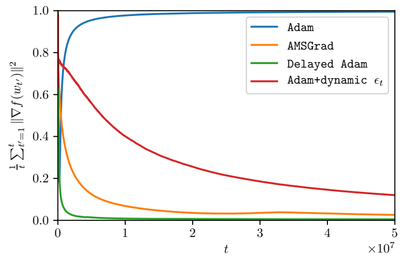
Figure 3: The mean gradient norm as function of the iteration t when optimizing Equation (11). Matching our theoretical results, Delayed Adam and Adam with dynamic ϵ_t both converge, while Adam fails to converge.

dynamics in both regimes: when ϵ is small, changing its value has negligible impact on η_t and $\|\eta_t\|$, hence the updates will be the same, while in the non-adaptive regime we have that $\eta_t \approx [\frac{1}{\epsilon}, \frac{1}{\epsilon}, \dots]$ and $\|\eta_t / \sqrt{d}\|_2 \approx \frac{1}{\epsilon}$, hence normalizing η_t yields an all-ones vector regardless of ϵ (as long as it remains large enough compared to all components of v_t).

Figure 1 shows the performance of a Wide ResNet 28-4 on CIFAR-10 when trained with Adam and AvaGrad, for different (α, ϵ) configurations *i.e.* the grid search employed in Section 5.1. For Adam, the non-adaptive regime is indeed characterized by a linear relation between α and ϵ, while its performance in the adaptive regime depends mostly on α alone. AvaGrad offers decoupling between the two parameters, with α precisely characterizing the non-adaptive regime (*i.e.* the performance is independent of ϵ) while almost fully describing the adaptive regime as well, except for regions close to the phase transition. For each of the 21 different values of ϵ, AvaGrad performed best with $\alpha = 1.0$.

7. Separability & Hyperparameter Tuning

To assess our hypothesis that AvaGrad offers hyperparameter decoupling, which enables α and ϵ to be tuned *independently* via two line-search procedures instead of a grid search, we compare tuning costs of Adam and AvaGrad with prominent hyperparameter optimization methods such as Parzen Trees and CMA-ES. We also consider Gradientless Descent (GLD) [10], a powerful zeroth-order method.

We frame the task of tuning α and ϵ as a 2D optimization problem with a 21×21 discrete domain representing all (α, ϵ) configurations explored in Section 5.1, with the minimization objective being the error of a Wide ResNet-28-4 on CIFAR-10 when trained with the corresponding (α, ϵ) values.

Figure 2 (left) shows the number of iterations required by different hyperparameter optimizers to achieve 0.5% suboptimality *i.e.* an error at most 0.5% higher than the lowest achieved in the grid. AvaGrad is significantly cheaper to tune than Adam, regardless of the adopted tuning algorithm, including random search – showing that AvaGrad is able to perform well with a wider range of hyperparameter values.

We also consider a variant of GLD where descent steps on α and ϵ are performed separately in an alternating manner, akin to coordinate descent [23, 32]. This variant, which we denote by CGLD, is in principle well-suited for problems where variables have independent contributions to the objective, as is approximately the case for AvaGrad. Results are given in Figure 2 (right): AvaGrad achieves less than 1% suboptimality in 13 iterations when tuned with CGLD, while Adam requires 74 with GLD. As expected, coordinate-wise updates result in considerably faster tuning for AvaGrad.

8. Theoretical Foundations

Finally, we present a theoretical analysis on the convergence of adaptive methods, but taking a different approach from the one considered in Section 3: instead of analyzing how ϵ affects the convergence of Adam, here we focus on better understanding *why* Adam can fail to converge for problems that depend on its hyperparameter settings.

We first note that the 'adversarial' problems designed to show Adam's nonconvergence in Theorem 3 of Reddi *et al.* [34] and our Theorem 1 exploit the correlation between m_t and η_t to guarantee that Adam takes overly conservative steps when presented with rare samples that contribute significant to the objective.

While Reddi *et al.* already propose a modification for Adam that guarantees its convergence, it relies on explicitly constraining the parameter-wise learning rates η_t to be pointwise decreasing which can harm the method's adaptiveness. We present a simple way to directly circumvent the fact that m_t and η_t are correlated without constraining η_t, guaranteeing a $\mathcal{O}(1/\sqrt{T})$ rate for stochastic non-convex problems while being applicable to virtually any adaptive method.

Our modification consists of employing a 1-step delay in the update of η_t, or equivalently replacing η_t by η_{t-1} in the method's update rule for w_{t+1}. Although there is still statistical dependence between m_t and v_t, this ensures that η_t is independent of the current sample s_t, which the following result shows to suffice for SGD-like convergence:

Theorem 3. *Assume that f is smooth and f_s has bounded gradients for all $s \in \mathcal{S}$. For any optimization method that performs updates following (4) such that η_t is independent of s_t and $L \leq \eta_{t,i} \leq H$ for positive constants L and H, setting $\alpha_t = \alpha' / \sqrt{T}$ yields*

$$\mathbb{E}\left[\|\nabla f(w_t)\|^2\right] \leq \mathcal{O}\left(\frac{1}{L\sqrt{T}}\left(\frac{1}{\alpha'} + \alpha' H^2\right)\right), \quad (8)$$

where w_t is uniformly sampled from $\{w_1, \dots, w_T\}$.

Moreover, if s_t is independent of $Z := \sum_{t=1}^T \alpha_t \min_i \eta_{t,i}$, then setting $\alpha_t = \alpha'_t / \sqrt{T}$ yields

$$\mathbb{E}\left[\|\nabla f(w_t)\|^2\right] \leq \mathcal{O}\left(\frac{1}{\sqrt{T}} \cdot \mathbb{E}\left[\frac{\sum_{t=1}^T 1 + {\alpha'_t}^2 \|\eta_t\|^2}{\sum_{t=1}^T \alpha'_t \min_i \eta_{t,i}}\right]\right), \quad (9)$$

where w_t is sampled from $p(t) \propto \alpha_t \cdot \min_i \eta_{t,i}$.

The bound in (9) depends on the learning rate α_t and on both the squared norm and smallest value of the parameter-wise learning rate η_t, namely $\|\eta_t\|^2$ and $\min_i \eta_{t,i}$, enabling us to analyze how the relation between α_t and η_t affects the convergence rate, including how the rate can be improved by adopting a learning rate α_t that depends on η_t.

Setting $\alpha'_t = \|\eta_t\|^{-1}$ yields a bound on the convergence rate of

$$\mathcal{O}\left(\frac{\sqrt{T}}{\sum_{t=1}^T \frac{\min_i \eta_{t,i}}{\|\eta_t\|}}\right) \quad (10)$$

Note that such bound is stronger than the one in (8): given constants L and H as in Theorem 3, we have $L \leq \min_i \eta_{t,i}$ and $\|\eta_t\| \leq \sqrt{d}H$, yielding an upper bound of $\mathcal{O}(H/L\sqrt{T})$ that matches (8) when $\alpha' = H^{-1}$.

Note that, for $\eta_t = 1/(\sqrt{v_t} + \epsilon)$, having $\alpha'_t = \|\eta_t\|^{-1}$ is equivalent to normalizing η_t prior to each update step, which is precisely how we arrived at AvaGrad's update rule (with the exception of accounting the d, the dimensionality of η_t, when performing normalization).

Additionally, Theorem 3 predicts the existence of two distinct regimes in the behavior of Adam-like methods. Taking $\alpha' = \Theta(H^{-1})$ minimizes the bound in (8) and yields a rate of $\mathcal{O}(\frac{1}{\sqrt{T}}\frac{H}{L}) = \mathcal{O}\left(\frac{1}{\sqrt{T}} \max\left(1, \frac{G_\infty}{\epsilon}\right)\right)$ once we take $L = (G_\infty + \epsilon)^{-1}$ and $H = \epsilon^{-1}$, which can be shown to satisfy $L \leq \eta_{t,i} \leq H$ for Adam-like methods.

In this case, the convergence rate depends on $\frac{G_\infty}{\epsilon}$, or, informally, how v_t compares to ϵ (G_∞ is an upper bound on the magnitude of the gradients, hence directly connected to v_t). This closely matches the empirical results presented in Figure 1, which shows two visible phases with a transition around $\epsilon = 10^{-3}$.

Lastly, we demonstrate our convergence results in Theorems 2 and 3 experimentally by employing Adam, AMSGrad, Adam with a 1-step delay (Delayed Adam), and Adam with $\epsilon_t = \sqrt{t^3}$, on a synthetic problem with the same form as the one used in the proof of Theorem 1:

$$\min_{w \in [0,1]} f(w), \qquad f_s(w) = \begin{cases} 999\frac{w^2}{2}, \text{ w.p. } \frac{1}{500} \\ -w, \text{ otherwise} \end{cases} \quad (11)$$

where $f(w) := \mathbb{E}[f_s(w)]$.

This problem admits a stationary point $w^\star \approx 0.5$, and satisfies Theorem 5 for $\beta_1 = 0, \beta_2 = 0.99, \epsilon = 10^{-8}$. Figure 3 presents $\frac{1}{t}\sum_{t'=1}^t \|\nabla f(w_{t'})\|^2$ during training, and shows that Adam fails to converge (Theorem 1), while both Delayed Adam and dynamic Adam converge successfully (Theorem 3 and Theorem 2). We attribute the faster convergence of Delayed Adam to the lack of constraints on the parameter-wise learning rates.

9. Conclusion

Adaptive methods are widely used when training complex architectures, but are far from being well-understood in theory and practice. Our theoretical results show that adaptive methods enjoy a SGD-like convergence under a constraint on how parameter-wise learning rates are computed from samples, motivating a simple and universal modification to provide convergence guarantees to arbitrary adaptive methods. Our analysis also suggests a sensible connection between the learning rate α, the adaptability parameter ϵ, and the magnitude of the stochastic gradients.

Experimentally, we show that, contrary to prior beliefs, adaptive methods can outperform SGD even on challenging tasks such as ImageNet – *given a large enough budget for hyperparameter tuning*. We identify hyperparameter tuning as a key concern in understanding and designing adaptive methods, and propose AvaGrad, a theoretically-motivated method that decouples α and ϵ.

AvaGrad enables cheap of tuning of α and ϵ with coordinate zeroth-order methods, requiring a fraction of time taken by other optimizers. Being able to outperform competing methods while offering efficient hyperparameter tuning, AvaGrad can be a valuable tool for practitioners with limited computational resources.

Acknowledgments. This work was supported in part by the University of Chicago CERES Center.

References

[1] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

[2] David Balduzzi, Marcus Frean, Lennox Leary, J. P. Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? *ICML*, 2017.

[3] Jinghui Chen, Dongruo Zhou, Yiqi Tang, Ziyan Yang, and Quanquan Gu. Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks. *arXiv:1806.06763*, 2018.

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *ECCV*, 2018.

[5] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. *ICLR*, 2019.

[6] Y. Dauphin, H. de Vries, J. Chung, and Y. Bengio. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *corrL*, 2015.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *ACL*, 2019.

[8] J. Duchi, E. Hazan, , and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *ICML*, 2011.

[9] Saeed Ghadimi and Guanghui Lan. Stochastic First- and Zeroth-order Methods for Nonconvex Stochastic Programming. *SIAM*, 2013.

[10] Daniel Golovin, John Karro, Greg Kochanski, Chansoo Lee, Xingyou Song, and Qiuyi Zhang. Gradientless Descent: High-Dimensional Zeroth-Order Optimization. *ICLR*, 2020.

[11] Sam Gross and Martin Wilber. Training and investigating residual nets. `https://github.com/facebook/fb.resnet.torch`, 2016.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *ECCV*, 2016.

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*, 2017.

[15] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen, 1991.

[16] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[18] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *CVPR*, 2017.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.

[20] Minguk Kang and Jaesik Park. ContraGAN: Contrastive Learning for Conditional Image Generation. *NeurIPS*, 2020.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[22] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[23] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. *ICML*, 2007.

[24] J. H. Lim and J. C. Ye. Geometric gan. *arXiv:1806.06763*, 2017.

[25] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *ICLR*, 2020.

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019.

[27] Liangchen Luo, Xiong, Yuanhao, Liu, Yan, and Xu. Sun. Adaptive gradient methods with dynamic bound of learning rate. *ICLR*, 2019.

[28] Jerry Ma and Denis Yarats. On the adequacy of untuned warmup for adaptive optimization. *AAAI*, 2021.

[29] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, 1994.

[30] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. An Analysis of Neural Language Modeling at Multiple Scales. *arXiv:1803.08240*, 2018.

[31] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Honza Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. *INTERSPEECH*, 2010.

[32] Yu. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM*, 2012.

[33] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*, 2016.

[34] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *ICLR*, 2018.

[35] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CVPR*, 2016.

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *IJCV*, 115(3), 2015.

[37] Pedro Savarese. On the Convergence of AdaBound and its Connection to SGD. *arXiv:1908.04457*, 2019.

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[39] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CVPR*, 2015.

[40] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and Efficient Object Detection. *arXiv:1911.09070*, 2019.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *NIPS*, 2017.

[42] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The Marginal Value of Adaptive Gradient Methods in Machine Learning. *NIPS*, 2017.

[43] Li Xiaoyu and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. *AISTATS*, 2019.

[44] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CVPR*, 2017.

[45] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *BMVC*, 2016.

[46] Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. *NIPS*, 2018.

[47] Zhiming Zhou, Qingru Zhang, Guansong Lu, Hongwei Wang, Weinan Zhang, and Yong Yu. Adashift: Decorrelation and convergence of adaptive learning rate methods. *ICLR*, 2019.

[48] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *NeurIPS*, 2020.