TAMING GANS WITH LOOKAHEAD–MINMAX

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative Adversarial Networks are notoriously challenging to train. The underlying minmax optimization is highly susceptible to the variance of the stochastic gradient and the rotational component of the associated game vector field. To tackle these challenges, we propose the Lookahead algorithm for minmax optimization, originally developed for single objective minimization only. The backtracking step of our Lookahead–minmax naturally handles the rotational game dynamics, a property which was identified to be key for enabling gradient ascent descent methods to converge on challenging examples often analyzed in the literature. Moreover, it implicitly handles high variance without using large mini-batches, known to be essential for reaching state of the art performance. Experimental results on MNIST, SVHN, CIFAR-10, and ImageNet demonstrate a clear advantage of combining Lookahead–minmax with Adam or extragradient, in terms of performance and improved stability, for negligible memory and computational cost. Using 30-fold fewer parameters and 16-fold smaller minibatches we outperform the reported performance of the class-dependent BigGAN on CIFAR-10 by obtaining FID of 12.19 *without* using the class labels, bringing state-of-the-art GAN training within reach of common computational resources.

1 INTRODUCTION

Gradient-based methods are the workhorse of machine learning. These methods optimize the parameters of a model with respect to a single objective $f\colon \mathcal{X} \to \mathbb{R}$. However, an increasing interest for multi-objective optimization arises in various domains—such as mathematics, economics, multi-agent reinforcement learning (Omidshafiei et al., 2017)—where several agents aim at optimizing their own cost function $f_i\colon \mathcal{X}_1 \times \cdots \times \mathcal{X}_N \to \mathbb{R}$ simultaneously.

A particularly successful class of algorithms of this kind are the Generative Adversarial Networks (Goodfellow et al., 2014, (GANs)), which consist of two players referred to as a generator and a discriminator. GANs were originally formulated as minmax optimization $f\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ (Von Neumann & Morgenstern, 1944), where the generator and the discriminator aim at minimizing and maximizing the same value function, see § 2. A natural generalization of gradient descent for minmax problems is the gradient descent ascent algorithm (GDA), which alternates between a gradient descent step for the min-player and a gradient ascent step for the max-player. This minmax training aims at finding a Nash equilibrium where no player has the incentive of changing its parameters.

Despite the impressive quality of the samples generated by the GANs—relative to classical maximum likelihood-based generative models—these models remain *notoriously difficult to train*. In particular, poor performance (sometimes manifesting as "mode collapse"), brittle dependency on hyperparameters, or divergence are often reported. Consequently, obtaining state-of-the-art performance was shown to require large computational resources (Brock et al., 2019), making well-performing models unavailable for common computational budgets.

It was recently empirically shown that: (i) GANs often converge to a locally stable stationary point that is *not* a differential Nash equilibrium (Berard et al., 2020); (ii) increased batch size improves GAN performances (Brock et al., 2019) in contrast to minimization (Defazio & Bottou, 2019; Shallue et al., 2018). A principal reason behind these differences is attributed to the *rotations* arising due to the adversarial component of the associated vector field of the gradient of the two player's parameters (Mescheder et al., 2018; Balduzzi et al., 2018), which are atypical for minimization. More precisely, the Jacobian of the associated vector field (see definition in § 2) can be decomposed into a

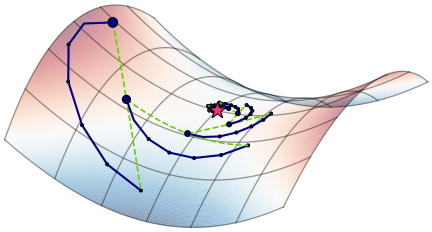


Figure 1: Illustration of Lookahead–minmax (Alg. 1) with GDA on: $\min_x \max_y x \cdot y$, with α=0.5. The solution, trajectory $\{x_t, y_t\}_{t=1}^{T}$, and the lines between $(x^{\mathcal{P}}, y^{\mathcal{P}})$ and (x_t, y_t) are shown with red star, blue line, and dashed green line, resp. The backtracking step of Alg. 1 (lines 10 & 11) allows the otherwise non-converging GDA to converge, see § 4.1.1.

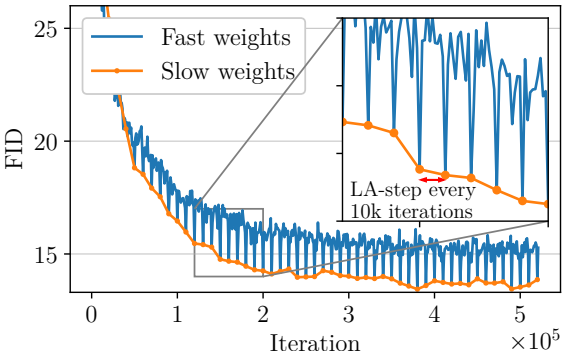


Figure 2: FID (↓) of Lookahead–minmax on 32×32 ImageNet (§ 5), see Fig. 9 for IS. We observe *significant* improvements after each LA-step, what empirically confirms the existence of rotations and thus the intuition of the behaviour of LA-minmax on games, as well as the relevance of the bilinear game for real world applications–Fig. 1.

symmetric and antisymmetric component (Balduzzi et al., 2018), which behave as a *potential* (Monderer & Shapley, 1996) and a *Hamiltonian* game, resp. Games are often combination of the two, making this general case hard to solve.

In the context of single objective minimization, Zhang et al. (2019) recently proposed the *Lookahead* algorithm, which intuitively uses an update direction by "looking ahead" at the sequence of parameters that change with higher variance due to the "stochasticity" of the gradient estimates–called *fast* weights–generated by an inner optimizer[1]. Lookahead was shown to improve the stability during training and to reduce the variance of the so called *slow* weights.

Contributions. Our contributions can be summarized as follows:

- Building on insights of Wang et al. (2020), who argue that Lookahead can be interpreted as an instance of local SGD, we derive improved convergence guarantees for the Lookahead algorithm in the context of single objective minimization.
- We propose Lookahead–minmax for optimizing minmax problems, that applies extrapolation in the *joint* parameter space (see Alg 1), so as to account for the rotational component of the associated game vector field (defined in § 2).
- We motivate the use of Lookahead–minmax for games by considering the extensively studied toy bilinear example (Goodfellow, 2016) and show that: (i) the use of lookahead allows for convergence of the otherwise diverging GDA on the classical bilinear game in full-batch setting (see § 4.1.1), as well as (ii) it yields good performance on challenging stochastic variants of this game, despite the high variance (see § 4.1.2).
- We empirically benchmark Lookahead–minmax on GANs on four standard datasets—MNIST, CIFAR-10, SVHN and ImageNet—*on two different models* (DCGAN & ResNet), with standard optimization methods for GANs, GDA and Extragradient, called LA–AltGAN and LA–ExtraGradient, resp. We *consistently* observe both stability and performance improvements at a negligible additional cost that does not require additional forward and backward passes, see § 5.

2 Background

GAN formulation. Given the data distribution p_d, the generator is a mapping $G\colon z \mapsto x$, where z is sampled from a known distribution $z \sim p_z$ and ideally $x \sim p_d$. The discriminator $D\colon x \mapsto D(x) \in [0,1]$ is a binary classifier whose output represents a conditional probability estimate that an x sampled from a balanced mixture of real data from p_d and G-generated data is

[1] As Lookahead uses iterates generated by inner optimizer we also refer it is a *meta-optimizer* or *wrapper*.

actually real. The optimization of a GAN is formulated as a differentiable two-player game where the generator G with parameters $\boldsymbol{\theta}$, and the discriminator D with parameters $\boldsymbol{\varphi}$, aim at minimizing their own cost function $\mathcal{L}^{\boldsymbol{\theta}}$ and $\mathcal{L}^{\boldsymbol{\varphi}}$, respectively, as follows:

$$\boldsymbol{\theta}^\star \in \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{L}^{\boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\varphi}^\star) \qquad \text{and} \qquad \boldsymbol{\varphi}^\star \in \arg\min_{\boldsymbol{\varphi}\in\Phi} \mathcal{L}^{\boldsymbol{\varphi}}(\boldsymbol{\theta}^\star, \boldsymbol{\varphi})\,. \tag{2P-G}$$

When $\mathcal{L}^{\boldsymbol{\varphi}} = -\mathcal{L}^{\boldsymbol{\theta}}$ the game is called a *zero-sum* and equation 2P-G is a minmax problem.

Minmax optimization methods. As GDA does *not* converge for some simple convex-concave game, Korpelevich (1976) proposed the *extragradient* method, where a "prediction" step is performed to obtain an extrapolated point $(\boldsymbol{\theta}_{t+\frac{1}{2}}, \boldsymbol{\varphi}_{t+\frac{1}{2}})$ using GDA, and the gradients at the *extrapolated* point are then applied to the current iterate $(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)$ as follows:

$$\text{Extrapolation:} \begin{cases} \boldsymbol{\theta}_{t+\frac{1}{2}} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\boldsymbol{\theta}}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t) \\ \boldsymbol{\varphi}_{t+\frac{1}{2}} = \boldsymbol{\varphi}_t - \eta \nabla_{\boldsymbol{\varphi}} \mathcal{L}^{\boldsymbol{\varphi}}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t) \end{cases} \text{Update:} \begin{cases} \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\boldsymbol{\theta}}(\boldsymbol{\theta}_{t+\frac{1}{2}}, \boldsymbol{\varphi}_{t+\frac{1}{2}}) \\ \boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t - \eta \nabla_{\boldsymbol{\varphi}} \mathcal{L}^{\boldsymbol{\varphi}}(\boldsymbol{\theta}_{t+\frac{1}{2}}, \boldsymbol{\varphi}_{t+\frac{1}{2}}) \end{cases} \tag{EG}$$

where η denotes the step size. In the context of zero-sum games, the extragradient method converges for any *convex-concave* function $\mathcal{L}$ and any closed convex sets Θ and Φ (Facchinei & Pang, 2003).

The joint vector field. Mescheder et al. (2017) and Balduzzi et al. (2018) argue that the vector field obtained by concatenating the gradients of the two players gives more insights of the dynamics than studying the loss surface. The joint vector field (JVF) and the Jacobian of JVF are defined as:

$$v(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \begin{pmatrix} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \\ \nabla_{\boldsymbol{\varphi}} \mathcal{L}^{\boldsymbol{\varphi}}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \end{pmatrix}, \text{ and } v'(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \begin{pmatrix} \nabla^2_{\boldsymbol{\theta}} \mathcal{L}^{\boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\varphi}) & \nabla_{\boldsymbol{\varphi}} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \\ \nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\varphi}} \mathcal{L}^{\boldsymbol{\varphi}}(\boldsymbol{\theta}, \boldsymbol{\varphi}) & \nabla^2_{\boldsymbol{\varphi}} \mathcal{L}^{\boldsymbol{\varphi}}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \end{pmatrix}, \text{ resp.} \tag{JVF}$$

Rotational component of the game vector field. Berard et al. (2020) show empirically that GANs converge to a *locally stable stationary point* (Verhulst, 1990, LSSP) that is not a differential Nash equilibrium–defined as a point where the norm of the Jacobian is zero and where the Hessian of both the players are *definite positive*. LSSP is defined as a point $(\boldsymbol{\theta}^\star, \boldsymbol{\varphi}^\star)$ where:

$$v(\boldsymbol{\theta}^\star, \boldsymbol{\varphi}^\star) = 0, \qquad \text{and} \qquad \mathcal{R}(\lambda) > 0, \forall \lambda \in Sp(v'(\boldsymbol{\theta}^\star, \boldsymbol{\varphi}^\star))\,, \tag{LSSP}$$

where $Sp(\cdot)$ denotes the spectrum of $v'(\cdot)$ and $\mathcal{R}(\cdot)$ the real part. In summary, (i) if all the eigenvalues of $v'(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)$ have positive real part the point $(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)$ is LSSP, and (ii) if the eigenvalues of $v'(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)$ have imaginary part, the dynamics of the game exhibit rotations.

Impact of noise due to the stochastic gradient estimates on games. Chavdarova et al. (2019) point out that relative to minimization, noise impedes more the game optimization, and show that there exists a class of zero-sum games for which the *stochastic* extragradient method *diverges*. Intuitively, bounded noise of the stochastic gradient hurts the convergence as with higher probability the noisy gradient points in a direction that makes the algorithm to diverge from the equilibrium, due to the properties of $v'(\cdot)$ (see Fig.1, Chavdarova et al., 2019).

3 LOOKAHEAD FOR SINGLE OBJECTIVE

In the context of single objective minimization, Zhang et al. (2019) recently proposed the Lookahead algorithm where at every step t: (i) a copy of the current iterate $\tilde{\boldsymbol{\omega}}_t$ is made: $\tilde{\boldsymbol{\omega}}_t \leftarrow \boldsymbol{\omega}_t$, (ii) $\tilde{\boldsymbol{\omega}}_t$ is then updated for $k \geq 1$ times yielding $\tilde{\boldsymbol{\omega}}_{t+k}$, and finally (iii) the actual update $\boldsymbol{\omega}_{t+1}$ is obtained as a *point that lies on a line between the two iterates*: the current $\boldsymbol{\omega}_t$ and the predicted one $\tilde{\boldsymbol{\omega}}_{t+k}$:

$$\boldsymbol{\omega}_{t+1} \leftarrow \boldsymbol{\omega}_t + \alpha(\tilde{\boldsymbol{\omega}}_{t+k} - \boldsymbol{\omega}_t), \qquad \text{where } \alpha \in [0, 1]\,. \tag{LA}$$

Lookahead uses two additional hyperparameters: (i) k–the number of steps used to obtain the prediction $\tilde{\boldsymbol{\omega}}_{t+k}$, as well as (ii) α– that controls how large a step we make towards the predicted iterate $\tilde{\boldsymbol{\omega}}$: the larger the closest, and when $\alpha = 1$ equation LA is equivalent to regular optimization (has no impact). Besides the extra hyperparameters, LA was shown to help the used optimizer to be more resilient to the choice of its hyperparameters, achieve faster convergence across different tasks, as well as to reduce the variance of the gradient estimates (Zhang et al., 2019).

Algorithm 1 General Lookahead–Minmax pseudocode.

Input: Stopping time T, learning rates $\eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{\varphi}}$, initial weights $\boldsymbol{\theta}_0$, $\boldsymbol{\varphi}_0$, lookahead k and α, losses $\mathcal{L}^{\boldsymbol{\theta}}$, $\mathcal{L}^{\boldsymbol{\varphi}}$, $\boldsymbol{d}^{\boldsymbol{\varphi}}_{(\cdot)}, \boldsymbol{d}^{\boldsymbol{\theta}}_{(\cdot)}$ base optimizer updates defined in § 4, p_d and p_z real and noise–data distributions, resp.
$\tilde{\boldsymbol{\theta}}_{0,0}, \tilde{\boldsymbol{\varphi}}_{0,0} \leftarrow \boldsymbol{\theta}_0, \boldsymbol{\varphi}_0$
for $t \in 0, \dots, T-1$ **do**
  for $i \in 0, \dots, k-1$ **do**
    Sample $\boldsymbol{x}, \boldsymbol{z} \sim p_d, p_z$
    $\tilde{\boldsymbol{\varphi}}_{t,i+1} = \tilde{\boldsymbol{\varphi}}_{t,i} - \eta_{\boldsymbol{\varphi}} \boldsymbol{d}^{\boldsymbol{\varphi}}_{t,i}(\tilde{\boldsymbol{\theta}}_{t,i}, \tilde{\boldsymbol{\varphi}}_{t,i}, \boldsymbol{x}, \boldsymbol{z})$
    Sample $\boldsymbol{z} \sim p_z$
    $\tilde{\boldsymbol{\theta}}_{t,i+1} = \tilde{\boldsymbol{\theta}}_{t,i+1} - \eta_{\boldsymbol{\theta}} \boldsymbol{d}^{\boldsymbol{\theta}}_{t,i}(\tilde{\boldsymbol{\theta}}_{t,i}, \tilde{\boldsymbol{\varphi}}_{t,i}, \boldsymbol{z})$
  end for
  $\boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t + \alpha_{\boldsymbol{\varphi}}(\tilde{\boldsymbol{\varphi}}_{t,k} - \boldsymbol{\varphi}_t)$
  $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_{\boldsymbol{\theta}}(\tilde{\boldsymbol{\theta}}_{t,k} - \boldsymbol{\theta}_t)$
  $\tilde{\boldsymbol{\theta}}_{t+1,0}, \tilde{\boldsymbol{\varphi}}_{t+1,0} \leftarrow \boldsymbol{\theta}_{t+1}, \boldsymbol{\varphi}_{t+1}$
end for
Output: $\boldsymbol{\theta}_T$, $\boldsymbol{\varphi}_T$

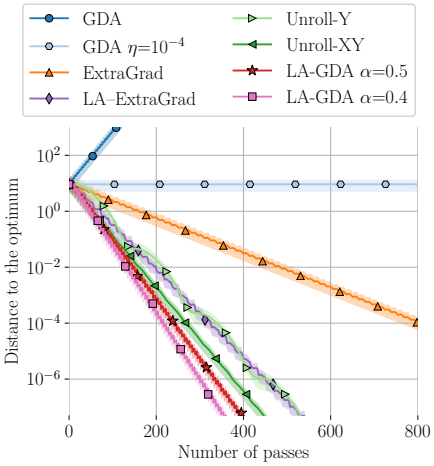


Figure 3: Distance to the optimum of equation SB–G using different *full–batch* methods, averaged over 5 runs. Unless otherwise specified, the learning rate is $\eta = 0.3$. See § 4.1.1.

Theoretical Analysis. The theoretical understanding of the LA optimizer (in the variant where k SGD steps are used for prediction) is still under development: Zhang et al. (2019) study the performance of LA on quadratic functions and Wang et al. (2020) recently provided an analysis for general smooth non-convex functions. One of their main observations is that LA can be viewed as an instance of local SGD (or parallel SGD, Stich, 2019; Koloskova et al., 2020; Woodworth et al., 2020b). Relying on the insights developed in these works, we can further improve upon the results by Wang et al. (2020), and we show in Appendix A that the number of gradient updates for converging to a stationary point $\mathbb{E}\|\nabla f(\boldsymbol{\omega})\|^2 \leq \varepsilon$, assuming stochastic gradient variance bounded by σ^2, can be bounded as:

$$\text{LA on smooth } f\text{: } \mathcal{O}\left(\frac{\sigma^2}{\varepsilon^2} + \frac{1}{\varepsilon} + \frac{1-\alpha}{\alpha}\left(\frac{\sigma\sqrt{k-1}}{\varepsilon^{3/2}} + \frac{k}{\varepsilon}\right)\right), \quad \begin{array}{l}\text{LA on quadratic } f\text{:}\\ \text{SGD on smooth } f\text{:}\end{array} \mathcal{O}\left(\frac{\sigma^2}{\varepsilon^2} + \frac{1}{\varepsilon}\right).$$

The asymptotically most significant term, $\mathcal{O}\left(\frac{\sigma^2}{\varepsilon^2}\right)$, of matches with the corresponding term in the SGD convergence rate for all choices of $\alpha \in (0, 1]$, and when $\alpha \to 1$, the same convergence guarantees as for SGD can be attained. When $\sigma^2 = 0$ the rate improves to $\mathcal{O}\left(\frac{1}{\epsilon}\right)$, in contrast to $\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$ in (Wang et al., 2020).[2] For small values of α, the worst-case complexity estimates of LA can in general be k times worse than for SGD (except for quadratic functions, where the rates match). Deriving tighter analyses for LA that corroborate the observed practical advantages is still an open problem.

4 Lookahead for Minmax Objectives

Although the simplistic bilinear game does *not* capture all the challenges of minmax optimization, it is instructive to consider it in order to understand the behavior of the optimizers around stationary points. In this bilinear game, taking a point on a line between two points on a cyclic trajectory would reduce the distance to the optimum—as illustrated in Fig. 1—what motivates the extension of Lookahead on games. Thus we use equation LA in joint parameter space $(\boldsymbol{\theta}, \boldsymbol{\varphi})$.

Alg. 1 summarizes the proposed Lookahead–minmax algorithm which for clarity does *not* cover *different update ratios* for the two players–see Alg. 3 for this case. At every step $t = 1, \dots, T$, we first compute k-step "predicted" iterates $(\tilde{\boldsymbol{\theta}}_{t,k}, \tilde{\boldsymbol{\varphi}}_{t,k})$—also called *fast* weights—using an inner "base" optimizer whose updates for the two players are denoted with $\boldsymbol{d}^{\boldsymbol{\varphi}}$ and $\boldsymbol{d}^{\boldsymbol{\theta}}$. For GDA and EG we have:

[2] Our results rely on optimally tuned stepsize γ for every choice of α. Wang et al. (2020) state a slightly different observation but keep the stepize γ fixed for different choices of α.

$$\text{GDA:} \begin{cases} \boldsymbol{d}^{\boldsymbol{\varphi}}_{t,i} \triangleq \nabla_{\tilde{\boldsymbol{\varphi}}} \mathcal{L}^{\boldsymbol{\varphi}}(\tilde{\boldsymbol{\theta}}_{t,i}, \tilde{\boldsymbol{\varphi}}_{t,i}, \boldsymbol{x}, \boldsymbol{z}) \\ \boldsymbol{d}^{\boldsymbol{\theta}}_{t,i} \triangleq \nabla_{\tilde{\boldsymbol{\theta}}} \mathcal{L}^{\boldsymbol{\theta}}(\tilde{\boldsymbol{\theta}}_{t,i}, \tilde{\boldsymbol{\varphi}}_{t,i}, \boldsymbol{z}) \end{cases} \qquad \text{EG:} \begin{cases} \boldsymbol{d}^{\boldsymbol{\varphi}}_{t,i} \triangleq \nabla_{\tilde{\boldsymbol{\varphi}}} \mathcal{L}^{\boldsymbol{\varphi}}(\tilde{\boldsymbol{\theta}}_{t,i+\frac{1}{2}}, \tilde{\boldsymbol{\varphi}}_{t,i+\frac{1}{2}}, \boldsymbol{x}, \boldsymbol{z}) \\ \boldsymbol{d}^{\boldsymbol{\theta}}_{t,i} \triangleq \nabla_{\tilde{\boldsymbol{\theta}}} \mathcal{L}^{\boldsymbol{\theta}}(\tilde{\boldsymbol{\theta}}_{t,i+\frac{1}{2}}, \tilde{\boldsymbol{\varphi}}_{t,i+\frac{1}{2}}, \boldsymbol{z}) \end{cases},$$

where for the latter, $\boldsymbol{\varphi}_{t,i+\frac{1}{2}}, \boldsymbol{\theta}_{t,i+\frac{1}{2}}$ are computed using equation EG. The above updates can be combined with *Adam* (Kingma & Ba, 2015), see § D for detailed descriptions of these algorithms. The so called *slow* weights $\boldsymbol{\theta}_{t+1}, \boldsymbol{\varphi}_{t+1}$ are then obtained as a point on a line between $(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)$ and $(\tilde{\boldsymbol{\theta}}_{t,k}, \tilde{\boldsymbol{\varphi}}_{t,k})$—*i.e.* equation LA, see lines 10 & 11 (herein called *backtracking* or LA–step). When combined with GDA, Alg. 1 can be *equivalently* re-written as Alg. 3, and note that it differs from using Lookahead separately per player–see § D.2. Moreover, Alg. 1 can be applied several times using *nested* LA–steps with (different) k, see Alg. 6.

4.1 Motivating example: the bilinear game

We argue that Lookahead-minmax allows for improved stability and performance on minmax problems due to two main reasons: (i) it handles well rotations, as well as (ii) it reduces the noise due to making more conservative steps. In the following, we disentangle the two, and show in § 4.1.1 that Lookahead-minmax converges fast in the full-batch setting, without presence of noise as each parameter update uses the full dataset. Moreover, besides that the GDA algorithm is known to diverge on this example, we show that when combined with lookahead it does converge.

In § 4.1.2 we consider the challenging problem of Chavdarova et al. (2019), designed to have high variance. We show that besides therein proposed *Stochastic Variance Reduced Extragradient* (SVRE), Lookahead-minmax is the only method that converges on this experiment, while considering all methods of Gidel et al. (2019a, §7.1). More precisely, we consider the following bilinear problem:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \max_{\boldsymbol{\varphi} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \min_{\boldsymbol{\theta} \in \mathbb{R}^d} \max_{\boldsymbol{\varphi} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{\theta}^\top \boldsymbol{b}_i + \boldsymbol{\theta}^\top \boldsymbol{A}_i \boldsymbol{\varphi} + \boldsymbol{c}_i^\top \boldsymbol{\varphi}), \tag{SB–G}$$

with $\boldsymbol{\theta}, \boldsymbol{\varphi}, \boldsymbol{b}_i, \boldsymbol{c}_i \in \mathbb{R}^d$ and $\boldsymbol{A} \in \mathbb{R}^{n \times d \times d}$, $n{=}d{=}100$, and we draw $[\boldsymbol{A}_i]_{kl} = \delta_{kli}$ and $[\boldsymbol{b}_i]_k, [\boldsymbol{c}_i]_k \sim \mathcal{N}(0, 1/d)$, $1 \leq k, l \leq d$, where $\delta_{kli}{=}1$ if $k{=}l{=}i$, and 0 otherwise. As one pass we count a forward and backward pass § B.4, and all results are normalized. See § B.1 for hyperparameters.

4.1.1 The full-batch setting

In Fig. 3 we compare: (i) *GDA* with learning rate $\eta = 10^{-4}$ and $\eta = 0.3$ (in blue), which oscilates around the optimum with small enough step size, and diverges otherwise; (ii) *Unroll-Y* where the max-player is unrolled k steps, before updating the min player, as in (Metz et al., 2017); (iii) *Unroll-XY* where both the players are unrolled k steps with fixed opponent, and the actual updates are done with un unrolled opponent (see § B); (iv) *LA–GDA* with $\alpha = 0.5$ and $\alpha = 0.4$ (in red and pink, resp.) which combines Alg. 1 with GDA. (v) *ExtraGradient*–Eq. EG; as well as (vi) *LA–ExtraGrad*, which combines Alg. 1 with ExtraGradient. See § B for description of the algorithms and their

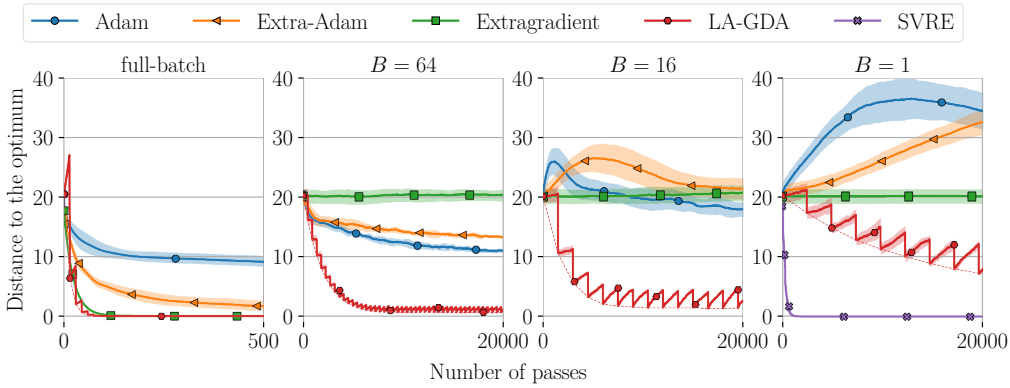


Figure 4: Convergence of *Adam*, *ExtraAdam*, *Extragradient*, *SVRE* and *LA-GDA*, on equation SB–G, for several minibatch sizes B, averaged over 5 runs–with random initialization of both the parameters and the data points ($\boldsymbol{A}$, $\boldsymbol{b}$ and $\boldsymbol{c}$). Fast $(\tilde{\boldsymbol{\varphi}}, \tilde{\boldsymbol{\theta}})$ and slow $(\boldsymbol{\varphi}, \boldsymbol{\theta})$ weights of LA–GDA are shown with solid and dashed lines, resp.

	Fréchet Inception distance ↓			Inception score ↑		
(32×32) **ImageNet**	no avg	EMA	EMA-slow	no avg	EMA	EMA-slow
AltGAN	$15.63 \pm .46$	$14.16 \pm .27$	–	$7.23 \pm .13$	$7.81 \pm .07$	–
LA–AltGAN	$14.37 \pm .20$	$\mathbf{13.06 \pm .20}$	$\mathbf{12.53 \pm .06}$	$7.58 \pm .07$	$\mathbf{7.97 \pm .11}$	$\mathbf{8.42 \pm .11}$
NLA–AltGAN	$\mathbf{13.14 \pm .25}$	$\mathbf{13.07 \pm .25}$	$12.71 \pm .13$	$\mathbf{7.85 \pm .05}$	$\mathbf{7.87 \pm .01}$	$8.10 \pm .05$
ExtraGrad	$15.48 \pm .44$	$14.15 \pm .63$	–	$7.31 \pm .06$	$7.85 \pm .10$	–
LA–ExtraGrad	$14.53 \pm .27$	$14.13 \pm .23$	$14.09 \pm .28$	$7.62 \pm .06$	$7.70 \pm .07$	$7.89 \pm .04$
NLA–ExtraGrad	$15.05 \pm .96$	$14.79 \pm .93$	$13.88 \pm .45$	$7.39 \pm .12$	$7.48 \pm .12$	$7.76 \pm .15$
CIFAR-10						
AltGAN	21.37 ± 1.60	16.92 ± 1.16	–	$7.41 \pm .16$	$8.03 \pm .13$	–
LA–AltGAN	$16.74 \pm .46$	$\mathbf{13.98 \pm .47}$	$\mathbf{12.67 \pm .57}$	$\mathbf{8.05 \pm .43}$	$\mathbf{8.19 \pm .05}$	$\mathbf{8.55 \pm .04}$
ExtraGrad	$18.49 \pm .99$	15.47 ± 1.82	–	$7.61 \pm .07$	$8.05 \pm .09$	–
LA–ExtraGrad	$\mathbf{15.25 \pm .30}$	$14.68 \pm .30$	$13.39 \pm .23$	$\mathbf{7.99 \pm .03}$	$8.04 \pm .04$	$8.40 \pm .05$
Unrolled–GAN	21.04 ± 1.08	17.51 ± 1.08	–	$7.43 \pm .07$	$7.88 \pm .12$	–
SVHN						
AltGAN	7.84 ± 1.21	6.83 ± 2.88	–	$3.10 \pm .09$	$\mathbf{3.19 \pm .09}$	–
LA–AltGAN	$3.87 \pm .09$	$\mathbf{3.28 \pm .09}$	$\mathbf{3.21 \pm .19}$	$3.16 \pm .02$	$\mathbf{3.22 \pm .08}$	$\mathbf{3.30 \pm .07}$
ExtraGrad	$4.08 \pm .11$	$\mathbf{3.22 \pm .09}$	–	$\mathbf{3.21 \pm .02}$	$\mathbf{3.16 \pm .02}$	–
LA–ExtraGrad	$\mathbf{3.20 \pm .09}$	$\mathbf{3.16 \pm .14}$	$\mathbf{3.15 \pm .31}$	$\mathbf{3.20 \pm .02}$	$\mathbf{3.19 \pm .03}$	$3.20 \pm .04$

Table 1: Comparison of LA–GAN with its respective baselines AltGAN and ExtraGrad (see § 5.1 for naming), using FID (lower is better) and IS (higher is better), and *best obtained scores*. EMA denotes *exponential moving average*, see § C. All methods use *Adam*, see § D for detailed description. Results are averaged over 5 runs. We run each experiment for 500K iterations. See § E and § 5.2 for details on architectures and hyperparameters and for discussion on the results, resp. Our overall best obtained FID scores are 12.19 on CIFAR-10 and 2.823 on SVHN, see § F for samples of these generators. Best scores obtained for each metric and dataset are highlighted in yellow. For each column the best score is in bold along with any score within its standard deviation reach.

	Unconditional GANs						Conditional GANs	
	SNGAN Miyato et al.	Prog.GAN Karras et al.	NCSN Song & Ermon	WS-SVRE Chavdarova et al.	ExtraAdam Gidel et al.	LA-AltGAN (ours)	SNGAN Miyato et al.	BigGAN Brock et al.
FID	21.7	–	25.32	16.77	$16.78 \pm .21$	**12.19**	25.5	14.73
IS	8.22	$8.80 \pm .05$	8.87	–	$8.47 \pm .10$	8.78	8.60	**9.22**

Table 2: Summary of the recently reported best scores on CIFAR-10 and benchmark with LA–GAN, *using published results*. Note that the architectures are *not* identical for all the methods–see § 5.2.

implementation. Interestingly, we observe that Lookahead–Minmax allows GDA to converge on this example, and moreover speeds up the convergence of ExtraGradient.

4.1.2 The stochastic setting

In this section, we show that besides SVRE, Lookahead–minmax also converges on equation SB–G. In addition, we test all the methods of Gidel et al. (2019a, §7.1) using minibatches of several sizes $B = 1, 16, 64$, and sampling without replacement. In particular, we tested: (i) the *Adam* method combined with GDA (shown in blue); (ii) *ExtraGradient*–Eq. EG; as well as (iii) *ExtraAdam* proposed by (Gidel et al., 2019a); (iv) our proposed method *LA-GDA* (Alg. 1) combined with GDA; as well as (v) *SVRE* (Chavdarova et al., 2019, Alg.1) for completeness. Fig. 4 depicts our results. See § B for details on the implementation and choice of hyperparameters. We observe that besides the good performance of LA-GDA on games in the batch setting, it also has the property to cope well large variance of the gradient estimates, and it converges *without* using restarting.

5 Experiments

In this section, we empirically benchmark Lookahead–minmax–Alg. 1 for *training GANs*. For the purpose of fair comparison, as an *iteration* we count each update of *both* the players, see Alg. 3.

5.1 EXPERIMENTAL SETUP

Datasets. We used the following image datasets: (i) **MNIST** (Lecun & Cortes, 1998), (ii) **CIFAR-10** (Krizhevsky, 2009, §3), (iii) **SVHN** (Netzer et al., 2011), and (iv) **ImageNet** ILSVRC 2012 (Russakovsky et al., 2015), using resolution of 28×28 for **MNIST**, and $3 \times 32 \times 32$ for the rest.

Metrics. We use **Inception score** (IS, Salimans et al., 2016) and **Fréchet Inception distance** (FID, Heusel et al., 2017) as these are most commonly used metrics for image synthesis, see § E.1 for details. On datasets other than ImageNet, IS is less consistent with the sample quality (see E.1.1).

DNN architectures. For experiments on **MNIST**, we used the DCGAN architectures (Radford et al., 2016), described in § E.2.1. For SVHN and CIFAR-10, we used the ResNet architectures, replicating the setup in (Miyato et al., 2018; Chavdarova et al., 2019), described in detail in E.2.2.

Optimization methods. We use: (i) **AltGan:** the standard alternating GAN, (ii) **ExtraGrad:** the extragradient method, as well as (iii) **UnrolledGAN:** (Metz et al., 2017). We combine Lookahead-minmax with (i) and (ii), and we refer to these as **LA–AltGAN** and **LA–ExtraGrad**, respectively or for both as **LA–GAN** for brevity. We denote *nested* LA–GAN with prefix **NLA**, see § D.1. All methods in this section use *Adam* (Kingma & Ba, 2015). We compute Exponential Moving Average (EMA, see def. in § C) of both the fast and slow weights–called **EMA** and **EMA–slow**, resp. See Appendix for results of uniform iterate averaging and *RAdam* (Liu et al., 2020).

5.2 RESULTS AND DISCUSSION

Comparison with baselines. Table 1 summarizes our comparison of combining Alg. 1 with AltGAN and ExtraGrad. On all datasets, we observe that the iterates (column "no avg") of LA–AltGAN and LA–ExtraGrad perform notably better than the corresponding baselines, and using EMA on LA–AltGAN and LA–ExtraGrad further improves the FID and IS scores obtained with LA–AltGAN. As the performances improve after each LA–step, see Fig. 2 computing EMA solely on the slow weights further improves the scores. It is interesting to note that on most of the datasets, the scores of the iterates of LA–GAN (column *no-avg*) outperform the EMA scores of the respective baselines. In some cases, EMA for AltGAN does not provide improvements, as the iterates diverge relatively early. In our baseline experiments, ExtraGrad outperforms AtlGAN–while requiring twice the computational time of the latter per iteration. The addition of Lookahead–minimax stabilizes AtlGAN making it competitive to LA–ExtraGrad while using *half* of the computational time.

In Table 3 we report our results on MNIST, where–different from the rest of the datasets–the training of the baselines is stable, to gain insights if LA–GAN still yields improved performances. The best FID scores of the iterates (column "no avg") are obtained with LA–GAN. Interestingly, although we obtain that the last iterates are not LSSP (which could be due to stochasticity), from Fig. 5–which depicts the eigenvalues of JVF, we observe that after convergence LA–GAN shows no rotations.

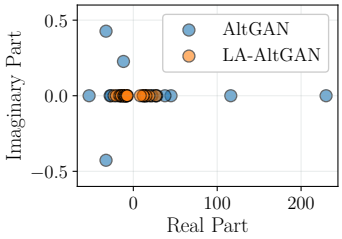
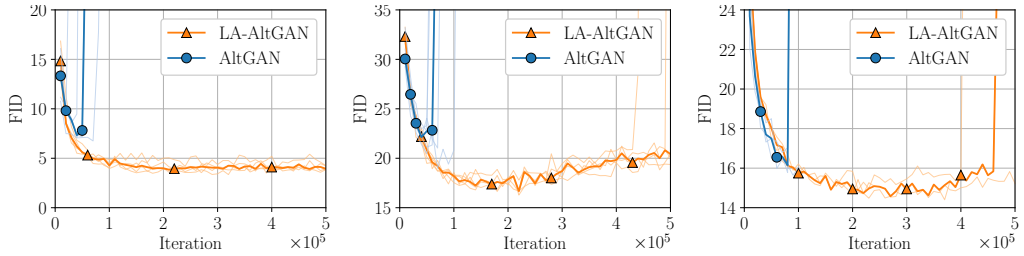


Figure 5: Eigenvalues of $v'(\boldsymbol{\theta}, \boldsymbol{\varphi})$ at 100K iterations on MNIST, see § 5.2.

	no avg	EMA	EMA-slow
AltGAN	$.094 \pm .006$	$\mathbf{.031 \pm .002}$	–
LA–AltGAN	$\mathbf{.053 \pm .004}$	$\mathbf{.029 \pm .002}$	$.032 \pm .002$
ExtraGrad	$.094 \pm .013$	$.032 \pm .003$	–
LA–ExtraGrad	$\mathbf{.053 \pm .005}$	$.032 \pm .002$	$.034 \pm .001$
Unrolled–GAN	$.077 \pm .006$	$\mathbf{.030 \pm .002}$	–

Table 3: FID (lower is better) results on MNIST, averaged over 5 runs. Each experiment is trained for 100K iterations. Note that Unrolled–GAN is computationally more expensive: in the order of the ratio $4:22$–as we used 20 steps of unrolling what gave best results. See 5.2 & E for implementation and discussion, resp.

Benchmark on CIFAR-10 using reported results. Table 2 summarizes the recently reported *best* obtained FID and IS scores on CIFAR-10. Although using the class labels–Conditional GAN is known to improve GAN performances (Radford et al., 2016), we outperform BigGAN (Brock et al., 2019) on CIFAR-10. Notably, our model and BigGAN have 5.1M and 158.3M parameters in total, respectively, and we use minibatch size of 128, whereas BigGAN uses 2048 samples. The competitive result of (Yazıcı et al., 2019) of average 12.56 FID uses ~ 3.5 times larger model than ours and is omitted from Table 2 as it does not use the standard metrics pipeline.



(a) SVHN dataset. (b) CIFAR-10 dataset. (c) ImageNet (32×32) dataset.

Figure 6: Improved stability of LA–AltGAN relative to its baselines on SVHN, CIFAR-10 and ImageNet, over 5 runs. The median and the individual runs are illustrated with ticker solid lines and with transparent lines, respectively. See § 5.2 for discussion.

Additional memory & computational cost. Lookahead-minmax requires the same extra memory footprint as EMA and uniform averaging (one additional set of parameters per player)—both of which are updated each step whereas LA–GAN is updated *once every* k iterations.

On the choice of α and k. We fixed $\alpha = 0.5$, and we experimented with several values of k–while once selected, keeping k fixed throughout the training. We observe that all values of k improve the baseline. Using larger k in large part depends on the stability of the base-optimizer: if it quickly diverges, smaller k is necessary to stabilize the training. Thus we used $k = 5$ and $k = 5000$ for LA–AltGAN and LA–ExtraGrad, resp. When using larger values of k we noticed that the obtained scores would periodically drastically improve every k iterations–after an LA step, see Fig.2. Interestingly, complementary to the results of (Berard et al., 2020)–who locally analyze the vector field, Fig.2 confirms the presence of rotation *while taking into account the used optimizer*, and empirically validates the geometric justification of Lookahead–minmax illustrated in Fig.1. The necessity of small k for unstable baselines motivates *nested* LA–Minmax using two different values of k, of which one is relatively small (the inner LA–step) and the other is larger (the outer LA–step). Intuitively, the inner and outer LA–steps in such nested LA–GAN tackle the variance and the rotations, resp.

Stability of convergence. Fig. 6 shows that LA–GAN *consistently improved the stability of its respective baseline*. Despite using $1 : 5$ update ratio for $G : D$–known to improve stability, our baselines always diverge (also reported by Chavdarova et al., 2019; Brock et al., 2019). On the other hand, LA–GAN diverged only few times and notably later in the training relative to the same baseline, see additional results in § F. The stability further improves using nested LA–steps as in Fig.2.

6 Related Work

Parameter averaging. In the context of convex single-objective optimization, taking an arithmetic average of the parameters as by Polyak & Juditsky (1992); Ruppert (1988) is well-known to yield faster convergence for convex functions and allowing the use of larger constant step-sizes in the case of stochastic optimization (Dieuleveut et al., 2017). It recently gained more interest in deep learning in general (Garipov et al., 2018), in natural language processing (Merity et al., 2018), and particularly in GANs (Yazıcı et al., 2019) where researchers report the performance of a uniform or exponential moving average of the iterates. Such averaging as a post-processing after training is fundamentally different from immediately applying averages during training. Lookahead as of our interest here in spirit is closer to extrapolation methods (Korpelevich, 1976) which rely on gradients taken not at the current iterate but at an extrapolated point for the current trajectory. For highly complex optimization landscapes such as in deep learning, the effect of using gradients at perturbations of the current iterate has a desirable smoothing effect which is known to help training speed and stability in the case of non-convex single-objective optimization (Wen et al., 2018; Haruki et al., 2019).

GANs. Several proposed methods for GANs are motivated by the "recurrent dynamics". Apart from the already introduced works, (i) Yadav et al. (2018) use prediction steps, (ii) Daskalakis et al. (2018) propose *Optimistic Mirror Decent* (OMD) for training Wasserstein GANs (Arjovsky et al., 2017), (iii) Balduzzi et al. (2018) propose the *Symplectic Gradient Adjustment* (SGA), among others. Besides its simplicity, the key benefit of LA–GAN is that it handles well *both* the rotations of the vector field as well as noise from stochasticity, thus performing well on real-world applications.

7 CONCLUSION

Motivated by the adversarial component of games and the negative impact of noise on games, we proposed an extension of the Lookahead algorithm to games, called "Lookahead–minmax". On the bilinear toy example we observe that combining Lookahead–minmax with standard gradient methods converges, and that Lookahead–minmax copes very well with the high variance of the gradient estimates. Exponential moving averaging of the iterates is known to help obtain improved performances for GANs, yet it does not impact the actual iterates, hence does not stop the algorithm from (early) divergence. Lookahead–minmax goes beyond such averaging, requires less computation than running averages, and it is *straightforward to implement*. It can be applied to any optimization method, and in practice it *consistently* improves the stability of its respective baseline. While we do *not* aim at obtaining a new state-of-the-art result, it is remarkable that Lookahead–minmax obtains competitive result on CIFAR–10 of 12.19 FID–outperforming the 30–times larger BigGAN.

As Lookahead–minmax uses two additional hyperparameters, future directions include developing adaptive schemes of obtaining these coefficients throughout training, which could further speed up the convergence of Lookahead–minmax.

REFERENCES

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. In *ICML*, 2018.

Hugo Berard, Gauthier Gidel, Amjad Almahairi, Pascal Vincent, and Simon Lacoste-Julien. A closer look at the optimization landscapes of generative adversarial networks. In *ICLR*, 2020.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

Ronald E Bruck. On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in hilbert space. *Journal of Mathematical Analysis and Applications*, 1977.

Tatjana Chavdarova, Gauthier Gidel, François Fleuret, and Simon Lacoste-Julien. Reducing noise in GAN training with variance reduced extragradient. In *NeurIPS*, 2019.

Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training GANs with optimism. In *ICLR*, 2018.

Aaron Defazio and Léon Bottou. On the ineffectiveness of variance reduced optimization for deep learning. In *NeurIPS*, 2019.

Aymeric Dieuleveut, Alain Durmus, and Francis Bach. Bridging the gap between constant step size stochastic gradient descent and markov chains. *arXiv:1707.06386*, 2017.

Francisco Facchinei and Jong-Shi Pang. *Finite-Dimensional Variational Inequalities and Complementarity Problems Vol I*. Springer Series in Operations Research and Financial Engineering, Finite-Dimensional Variational Inequalities and Complementarity Problems. Springer-Verlag, 2003.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *NeurIPS*, 2018.

Gauthier Gidel, Hugo Berard, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial nets. In *ICLR*, 2019a.

Gauthier Gidel, Reyhane Askari Hemmat, Mohammad Pezeshki, Rémi Le Priol, Gabriel Huang, Simon Lacoste-Julien, and Ioannis Mitliagkas. Negative momentum for improved game dynamics. In *AISTATS*, 2019b.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv:1701.00160*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

Kosuke Haruki, Taiji Suzuki, Yohei Hamakawa, Takeshi Toda, Ryuji Sakai, Masahiro Ozawa, and Mitsuhiro Kimura. Gradient Noise Convolution (GNC): Smoothing Loss Function for Distributed Large-Batch SGD. *arXiv:1906.10822*, 2019.

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

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. In *NIPS*, 2017.

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

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

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

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian U. Stich. A unified theory of decentralized SGD with changing topology and local updates. *ICML*, 2020.

GM Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 1976.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, 2009.

Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits. 1998. URL http://yann.lecun.com/exdb/mnist/.

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. In *ICLR*, 2020.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing LSTM language models. In *ICLR*, 2018.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of GANs. In *NIPS*, 2017.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which Training Methods for GANs do actually Converge? In *ICML*, 2018.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 1996.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011. URL http://ufldl.stanford.edu/housenumbers/.

Shayegan Omidshafiei, Jason Pazis, Chris Amato, Jonathan P. How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *ICML*, 2017.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 1992. doi: 10.1137/0330046.

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

David Ruppert. Efficient estimations from a slowly convergent Robbins-Monro process. *Technical report, Cornell University Operations Research and Industrial Engineering*, 1988.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *NIPS*, 2016.

Christopher J Shallue, Jaehoon Lee, Joe Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *arXiv:1811.03600*, 2018.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, pp. 11895–11907, 2019.

Sebastian U. Stich. Local SGD converges fast and communicates little. *ICLR*, 2019.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *arXiv:1512.00567*, 2015.

Ferdinand Verhulst. *Nonlinear Differential Equations and Dynamical Systems*. 1990.

J Von Neumann and O Morgenstern. *Theory of games and economic behavior.* Princeton University Press, 1944.

J. Wang, V. Tantia, N. Ballas, and M. Rabbat. Lookahead converges to stationary points of smooth non-convex functions. In *ICASSP*, 2020.

Wei Wen, Yandan Wang, Feng Yan, Cong Xu, Chunpeng Wu, Yiran Chen, and Hai Li. Smoothout: Smoothing out sharp minima to improve generalization in deep learning. *arXiv:1805.07898*, 2018.

Blake Woodworth, Kumar Kshitij Patel, and Nathan Srebro. Minibatch vs local SGD for heterogeneous distributed learning. *arXiv*, 2020a.

Blake Woodworth, Kumar Kshitij Patel, Sebastian U. Stich, Zhen Dai, Brian Bullins, H. Brendan McMahan, Ohad Shamir, and Nathan Srebro. Is local SGD better than minibatch SGD? *ICML*, 2020b.

Abhay Yadav, Sohil Shah, Zheng Xu, David Jacobs, and Tom Goldstein. Stabilizing adversarial nets with prediction methods. In *ICLR*, 2018.

Yasin Yazıcı, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in GAN training. In *ICLR*, 2019.

Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *NeurIPS*, 2019.