Reducing Noise in GAN Training with Variance Reduced Extragradient

Tatjana Chavdarova [*1 2] **Gauthier Gidel** [*1] **François Fleuret** [2] **Simon Lacoste-Julien** [1 3]

Abstract

Using large mini-batches when training generative adversarial networks (GANs) has been recently shown to significantly improve the quality of the generated samples. This can be seen as a simple but computationally expensive way of reducing the noise of the gradient estimates. In this paper, we investigate the effect of the noise in this context and show that it can prevent the convergence of standard stochastic game optimization methods, while their respective batch version converges. To address this issue, we propose a variance-reduced version of the stochastic extragradient algorithm (SVRE). We show experimentally that it performs similarly to a batch method, while being computationally cheaper, and show its theoretical convergence, improving upon the best rates proposed in the literature. Experiments on several datasets show that SVRE improves over baselines. Notably, SVRE is the first optimization method for GANs to our knowledge that can produce near state-of-the-art results *without* using adaptive step-size such as Adam.

1. Introduction

The current success of large-scale machine learning algorithms in large part relies on incremental gradient-based optimization methods to minimize empirical losses. These iterative methods handle large training datasets by computing estimates of the said gradient on mini-batches instead of using all the samples at every step, resulting in a method called *stochastic gradient descent* (SGD) (Robbins and Monro, 1951; Bottou, 2010).

While this method is reliably very efficient for classical loss minimization, such as cross-entropy for classification or squared loss for regression, recent works go beyond that setup, and aim at making several agents interact together with competing objectives. The associated optimization paradigm requires a multi-objective joint minimization.

One very popular class of models in that family are the generative adversarial networks (GANs, Goodfellow et al., 2014), which aim at finding a Nash equilibrium of a two-player *minimax* game, where the players are deep neural networks (DNNs).

1.1. Failure of SGD on multi-objective problems

Due to their success on supervised tasks, SGD based algorithms have been adopted for GAN training as well. However, convergence failures, poor performance (sometimes referred to as "mode collapse"), or hyperparameter susceptibility are much more commonly reported compared to classical supervised DNN optimization.

Recent works (Li et al., 2018; Gidel et al., 2019) argue that the currently used first-order methods may fail to converge on simple examples. Gidel et al. (2019) proposed instead to use an optimization technique coming from the variational inequality literature called *extragradient* (Korpelevich, 1976) with provable convergence guarantees to optimize games. However, we argue that the multi-objective minimization formulation of GANs introduces new challenges in terms of optimization. For example, we point out that the noise due to the stochasticity may break standard optimization techniques for stochastic games such as the stochastic extragradient method, by providing an example of stochastic bilinear game for which it provably does not converge.

This theoretical consideration is further supported empirically by the fact that using larger mini-batch sizes for GAN training has been shown to considerably improve the quality of the produced samples of the resulting generative model. More precisely, Brock et al. (2019) report a relative improvement of 46% of the Inception Score metric (see § 5) on ImageNet if the batch size is increased 8–fold. Nevertheless, this comes at the cost of an increase in the required computational budget that is prohibitively expensive for most of academic researchers in the machine learning community.

[*]Equal contribution [1]Mila & DIRO, Université de Montréal [2]École Polytechnique Fédérale de Lausanne and Idiap Research Institute [3]CIFAR fellow, Canada CIFAR AI chair. Correspondence to: Tatjana Chavdarova and Gauthier Gidel <firstname.lastname@umontreal.ca>.

Preprint. Work in progress.

Method	Complexity	μ-adaptivity
SVRG	$\ln(1/\epsilon) \times (n + \frac{\bar{L}^2}{\mu^2})$	✗
Acc. SVRG	$\ln(1/\epsilon) \times (n + \sqrt{n}\frac{\bar{L}}{\mu})$	✗
SVRE	$\ln(1/\epsilon) \times (n + \frac{\bar{L}}{\mu})$	✓

Table 1: Comparison of SVRE (Alg. 1) with other variance reduced methods for games for a μ-strongly monotone operator with L_i-Lipschitz stochastic estimators (see (10) for the definition of $\bar{L}$). We consider that the computation of one stochastic gradient is $O(1)$. SVRG and Accelerated SVRG *for monotone operators* (a generalization of games) have been proposed by Palaniappan and Bach (2016, Thm. 2 & Thm. 3).[1] The column *μ-adaptivity* indicates whether the algorithm's hyper-parameters (step size & epoch length) that guarantee convergence depend on the strong monotonicity parameter μ: if not, then the algorithm is adaptive to local strong monotonicity, a useful property to apply these results in the non-convex setting (see discussion after Thm. 3).

1.2. Our contributions

In this paper, we investigate the interplay between noise and multi-objective problems in the context of GAN training, and propose the novel "stochastic variance reduced extragradient" (SVRE) algorithm.

Our contributions can be summarized as follows:

- We show in a motivating example how the noise can make standard stochastic extragradient fail (see § 3.1).
- We propose a new method that combines variance reduction and extrapolation (SVRE) and show experimentally that SVRE effectively reduces the noise on real-world datasets (see § 4.2).
- We prove the convergence of SVRE under local strong convexity assumptions, improving over the best rates proposed in the literature (see § 4.2 and Table 1).
- We demonstrate experimentally the performance of SVRE to train GANs on MNIST, SVHN, ImageNet, and CIFAR-10 with *fixed* step-sizes. To our knowledge, SVRE is the first optimization method that can produce near state-of-the-art results without using adaptive step-size such as Adam (Kingma and Ba, 2015) (see § 5).

2. GANs as a game

The models/players in a GAN are respectively a generator G, that maps an embedding space to the signal space, and should eventually map a fixed noise distribution to the training data distribution, and a discriminator D whose only role is to allow the training of the generator by classifying genuine samples against generated ones. The central idea is that as long as D is able to do better than random, G is not properly modeling the data.

In this algorithm, at each iteration of the stochastic gradient descent (SGD), the discriminator D is updated to improve its "real vs. generated" classification performance, and the generator G is updated to degrade it.

Game theory formulation of GANs From a game theory point of view, GAN training is a differentiable two-player game: the discriminator $D_{\boldsymbol{\varphi}}$ aims at minimizing its cost function $\mathcal{L}^{(\boldsymbol{\varphi})}$ and the generator $G_{\boldsymbol{\theta}}$ aims at minimizing its own cost function $\mathcal{L}^{(\boldsymbol{\theta})}$. Using the same formulation as the one in Mescheder et al. (2017) and Gidel et al. (2019), the GAN objective has the following form,

$$\begin{cases} \boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\Theta} \mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\theta}, \boldsymbol{\varphi}^*) \\ \boldsymbol{\varphi}^* \in \arg\min_{\boldsymbol{\varphi}\in\Phi} \mathcal{L}^{(\boldsymbol{\varphi})}(\boldsymbol{\theta}^*, \boldsymbol{\varphi}) \,. \end{cases} \tag{2P-G}$$

When $\mathcal{L}^{(\boldsymbol{\varphi})} = -\mathcal{L}^{(\boldsymbol{\theta})} =: \mathcal{L}$ this game is called a *zero-sum game* and (2P-G) can be formulated as a minimax problem:

$$\min_{\boldsymbol{\theta}\in\Theta} \max_{\boldsymbol{\varphi}\in\Phi} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \tag{SP}$$

The gradient method is known to not converge for some convex-concave examples (Gidel et al., 2019). To fix this issue, Korpelevich (1976) proposed to use the *extragradient* method[2]:

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

This method performs a lookahead step in order to get signal from an extrapolated point, damping the oscillations.

It can be shown that, in the context of a zero-sum game, for any *convex-concave* function $\mathcal{L}$ and any closed convex sets Θ and Φ, the extragradient method does converge. We give below such a convergence result without rates for simplicity.

Theorem 1 (Theorem 12.1.11, Harker and Pang, 1990)**.** *Let Θ and Φ be two closed convex sets, let $\mathcal{L}$ a L-smooth function such that $\mathcal{L}(\cdot, \boldsymbol{\varphi})$ be convex for any $\boldsymbol{\varphi} \in \Phi$ and let*

[1] Note that the complexity results expressed in their Table 1 focus on a more specific bilinear problem analyzed in terms of the row and column-dimension of a matrix (the product of these dimensions correspond to our n). We focus instead on the general finite sum setting, using their Thm. 2 & 3.

[2] For simplicity of presentation, we describe the algorithms for the *unconstrained* setting where $\Theta = \mathbb{R}^d$. In the *constrained* scenario, a Euclidean projection on the constraint set should be added at every update of the extragradient method. This more general version (25) is the one analyzed in the appendix.

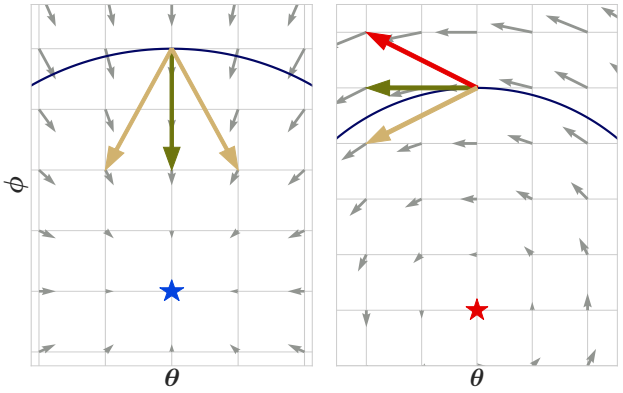


Figure 1: Illustration of the discrepancy between games and minimization on simple examples (1). **Left: minimization**. Up to a neighborhood, the noisy gradient always points to a direction that make the iterate closer to the minimum ($\star$). **Right: game**. The noisy gradient may point to a direction (red arrow) that push the iterate away from the Nash Equilibrium ($\star$).

$\mathcal{L}(\boldsymbol{\theta}, \cdot)$ be concave for any $\boldsymbol{\theta} \in \Theta$. If $\eta < 1/L$, then the sequence $(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)$ generated by the extragradient method (EG) *converges to a solution of* (SP).

3. Noise in Games

In standard large scale machine learning applications such as GANs, one cannot afford to compute the full batch gradient of the objective function at each time-step because of the too large batch size. The standard way to cope with this issue is to sample mini-batches of reasonable size and to use the gradient estimated on each mini-batch as an *unbiased estimator* of the "full" gradient. Unfortunately, the resulting noise in the gradient estimate may interact with the oscillations due to the adversarial component of the game.

We illustrate this phenomenon in Fig. 1 by contrasting the direction given by the noisy gradient on the following game and minimization problem, respectively:

$$\textit{game: } \min_{\theta \in \mathbb{R}} \max_{\phi \in \mathbb{R}} \theta \cdot \phi \,, \qquad \textit{min: } \min_{\theta, \phi \in \mathbb{R}} \theta^2 + \phi^2 \,. \tag{1}$$

Since the batch version of the gradient methods (updating both players simultaneously or alternatively) fails to converge for some convex (bilinear) games, there is no hope of convergence for their respective stochastic version. However, since (EG) does converge (Thm. 1), we could reasonably expect that its stochastic version does too (at least to a neighborhood). In the following section, we show that even under reasonable assumption this assertion is false: We present a simple example where the extragradient method does *converge linearly* (Gidel et al., 2019, Corollary 1) using the full gradient but diverges geometrically when using a stochastic estimate of it.[3]

[3] On this example, standard gradient methods diverge.

3.1. Stochasticity Breaks Extragradient

In the following, we show that, (a) on one hand, if we use standard stochastic estimates of the gradients of $\mathcal{L}$ with a simple finite sum formulation, then the iterates $\boldsymbol{\omega}_t := (\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)$ produced by the stochastic extragradient method (SEG) diverge geometrically, and (b) on the other hand, Theorem 1 ensures that the full-batch extragradient method does converge to the Nash equilibrium $\boldsymbol{\omega}^*$ of this game. All detailed proofs can be found in § B.

Theorem 2 (Noise may induce divergence)**.** *There exists a zero-sum stochastic game such that if $\boldsymbol{\omega}_0 \neq \boldsymbol{\omega}^*$, then for any step-size $\eta > 0$, the iterates $(\boldsymbol{\omega}_t)$ computed by the stochastic extragradient method diverge geometrically, i.e., there exists $\rho > 0$, such that $\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|^2] > \|\boldsymbol{\omega}_0 - \boldsymbol{\omega}^*\|^2 (1+\rho)^t$.*

Proof sketch. We consider the following stochastic optimization problem,

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \max_{\boldsymbol{\varphi} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\theta}^\top \boldsymbol{A}_i \boldsymbol{\varphi} \tag{2}$$

where $[\boldsymbol{A}_i]_{kl} = 1$ if $k = l = i$ and 0 otherwise. Note that this problem is a simple dot product between $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$, thus we can compute the batch gradient and notice that the Nash equilibrium of this problem is $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*) = (\mathbf{0}, \mathbf{0})$. However, as we will see below, this simple problem actually breaks standard stochastic optimization methods.

Sampling a mini-batch without replacement $I \subset \{1, \dots, n\}$, we denote $\boldsymbol{A}_I := \sum_{i \in I} \boldsymbol{A}_i$. The extragradient update rule can be written as

$$\begin{cases} \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{A}_I (\boldsymbol{\varphi}_t + \eta \boldsymbol{A}_J \boldsymbol{\theta}_t) \\ \boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t + \eta \boldsymbol{A}_I (\boldsymbol{\theta}_t - \eta \boldsymbol{A}_J \boldsymbol{\varphi}_t) \end{cases} \tag{3}$$

where I and J are the mini-batches respectively sampled for the update and the extrapolation step. Let us write $N_t := \|\boldsymbol{\theta}_t\|^2 + \|\boldsymbol{\varphi}_t\|^2$. Noticing that $[\boldsymbol{A}_I \boldsymbol{\theta}]_i = [\boldsymbol{\theta}]_i$ if $i \in I$ and 0 otherwise, we have,

$$\mathbb{E}[N_{t+1} | \boldsymbol{\theta}_t, \boldsymbol{\varphi}_t] = \left(1 - \eta^2 \left(2\tfrac{|I|^2}{n^2} - \tfrac{|I|}{n} + \tfrac{|I|^2}{n^2}\eta^2\right)\right) N_t$$

Consequently, if the mini-batch size is smaller than half of the dataset size, i.e. $2|I| \leq n$, we have that $\forall \eta > 0\,,\ \exists \rho > 0\,,\ s.t.\,,\ \mathbb{E}[N_t] > N_0 (1+\rho)^t$. □

This result may seem contradictory with the standard result on SEG (Juditsky et al., 2011) saying that the average of the iterates computed by SEG does converge to the Nash equilibrium of the game. But one fundamental assumption made by Juditsky et al. is that the estimator of the gradient has a finite variance. This assumption breaks in this example since the variance of the estimator is proportional to the norm of the parameters.

Thus, constraining the optimization problem (2) to bounded domains Θ and Φ,

$$\min_{\boldsymbol{\theta}\in\Theta}\max_{\boldsymbol{\varphi}\in\Phi}\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\theta}^\top \boldsymbol{A}_i\boldsymbol{\varphi} \tag{4}$$

would make the finite variance assumption from Juditsky et al. (2011) hold. Consequently, the averaged iterate $\bar{\boldsymbol{\omega}}_t := \frac{1}{t}\sum_{s=0}^{t-1}\boldsymbol{\omega}_s$ would converge to $\boldsymbol{\omega}^*$. However we argue in the next section that such result may not be satisfying for *non-convex* problems.

3.2. Why is convergence of the last iterate preferable?

In the light of Theorem 2, the behavior of the iterates on the constrained problem (4) is the following: they will diverge until they reach the boundary of Θ and Φ and then they will start to turn around the Nash equilibrium of (4) lying on these boundaries. Using convexity properties, we can then show that the averaged iterates will converge to the Nash equilibrium of the problem. However, with an arbitrary large domain, this convergence rate may be arbitrary slow (since it depends on the diameter of the domain).

Moreover, this behavior might be even more problematic in a non-convex framework because even if by chance we initialize close to the Nash equilibrium, we would get away from it and we cannot rely on convexity to expect the average of the iterates to converge.

Consequently, we would like optimization algorithms generating iterates that *stay close to the Nash equilibrium*.

4. Reducing Noise with VR Methods

One straightforward way to reduce the noise in the estimation of the gradient is to use mini-batches of samples instead of just one sample. However, mini-batches stochastic extragradient fails to converge on (4) if the mini-batch size is smaller than half of the dataset size (see § B.1 for more details). In order to get an estimator of the gradient with a vanishing variance, the optimization literature proposed to take advantage of the finite-sum formulation that often appears in machine learning (Schmidt et al., 2017, and references therein).

4.1. Variance Reduced Methods

Motivated by the GAN setup, let us assume that the objective in (2P-G) can be decomposed as a finite sum such that,

$$\mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\omega})=\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_i^{(\boldsymbol{\theta})}(\boldsymbol{\omega}),\quad \mathcal{L}^{(\boldsymbol{\varphi})}(\boldsymbol{\omega})=\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_i^{(\boldsymbol{\varphi})}(\boldsymbol{\omega}) \tag{5}$$

where we denote $\boldsymbol{\omega} := (\boldsymbol{\theta}, \boldsymbol{\varphi})$.

Johnson and Zhang (2013) propose the "stochastic variance reduced gradient" (SVRG) as an *unbiased* estimator of the gradient with a smaller variance than the vanilla mini-batch estimate. The idea is to occasionally take a snapshot $\boldsymbol{\omega}^{\mathcal{S}}$ of the current model's parameters, and store the full gradient $\boldsymbol{\mu}^{\mathcal{S}}$ at this point. Computing the full gradient at $\boldsymbol{\omega}^{\mathcal{S}}$ is an expensive operation but not prohibitive if it is done infrequently enough (for instance once every pass).

Assuming that we have stored $\boldsymbol{\omega}^{\mathcal{S}}$ and $\boldsymbol{\mu}^{\mathcal{S}} := (\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\mathcal{S}}, \boldsymbol{\mu}_{\boldsymbol{\varphi}}^{\mathcal{S}})$, *unbiased* estimates of the respective gradient of $\mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\omega})$ and $\mathcal{L}^{(\boldsymbol{\varphi})}(\boldsymbol{\omega})$ are,

$$\boldsymbol{d}_i^{(\boldsymbol{\theta})}(\boldsymbol{\omega}) := \frac{1}{n\pi_i}\left(\nabla\mathcal{L}_i^{(\boldsymbol{\theta})}(\boldsymbol{\omega}) - \nabla\mathcal{L}_i^{(\boldsymbol{\theta})}\left(\boldsymbol{\omega}^{\mathcal{S}}\right)\right) + \boldsymbol{\mu}_{\boldsymbol{\theta}}^{\mathcal{S}} \tag{6}$$

$$\boldsymbol{d}_i^{(\boldsymbol{\varphi})}(\boldsymbol{\omega}) := \frac{1}{n\pi_i}\left(\nabla\mathcal{L}_i^{(\boldsymbol{\varphi})}(\boldsymbol{\omega}) - \nabla\mathcal{L}_i^{(\boldsymbol{\varphi})}\left(\boldsymbol{\omega}^{\mathcal{S}}\right)\right) + \boldsymbol{\mu}_{\boldsymbol{\varphi}}^{\mathcal{S}}. \tag{7}$$

Actually, $\mathbb{E}[\boldsymbol{d}_i^{(\boldsymbol{\theta})}(\boldsymbol{\omega})] = \frac{1}{n}\sum_{i=1}^{n}\nabla\mathcal{L}_i^{(\boldsymbol{\theta})}(\boldsymbol{\omega}) = \nabla\mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\omega})$, where the expectation is taken over i, picked with a probability π_i. We call this unbiased estimate of the gradient the *SVRG estimate*. SVRG is an *epoch based* algorithm where an epoch is the inner loop which updates incrementally the parameters using the SVRG estimate between two snapshots which are updated in the outer loop.

The non-uniform sampling probabilities π_i are used to balance samples in spite of large variations of gradient estimates Lipschitz constant. This strategy has been first introduced for variance reduced methods by Xiao and Zhang (2014) for SVRG, and has been discussed for saddle point optimization by Palaniappan and Bach (2016).

Originally, SVRG was introduced as an epoch based algorithm with a *fixed epoch size*: in Alg. 1, one epoch is an inner loop of size N (Line 7). However, Hofmann et al. (2015) proposed instead to *sample* the size of each epoch from a geometric distribution. Doing so Hofmann et al. defined a notion of q-memorization algorithm that unifies SAGA, SVRG and other variants of variance reduction for incremental gradient methods with similar convergence rates. We generalize their notion of q-memorization algorithm to handle the extrapolation step (EG) and provide a convergence proof for such q-memorization algorithms in § B.2.

One advantage of Hofmann et al. (2015)'s framework is also that the sampling of the epoch size does not depend on the condition number of the problem, whereas the original proof for SVRG had to consider an epoch size larger than the condition number (see Leblond et al. (2018, Corollary 16) for a detailed discussion on the convergence rate for SVRG). Thus, this new version of SVRG with a random epoch size becomes *adaptive to the local strong convexity* since none of its hyper-parameters depend on the strong convexity constant.

However, because of some technical aspects introduced with monotone operators, Palaniappan and Bach (2016)'s proofs (both for SAGA and SVRG) require a step-size that

depends on the strong monotonicity constant making these algorithms not adaptive to local strong monotonicity. This motivates the proposed SVRE algorithm, which is adaptive to local strong monotonicity, and is thus more appropriate for non-convex optimization.

4.2. SVRE: Variance Reduced Extragradient

We describe our proposed algorithm called stochastic variance reduced extragradient (SVRE) in Alg. 1. In an analogous manner that Palaniappan and Bach (2016) combined SVRG with the gradient method to solve games, SVRE combines SVRG estimates of the gradient (6-7) with the *extragradient method* (EG). While the algorithmic proposal is simple, the proof of convergence is non-trivial. Moreover, with this method we are able to improve the best known convergence rates for variance reduction method for stochastic games (Table 1 and Thm. 3), and we show in § 4.3 that it is the only method which empirically converges on the simple example of § 3.1. We now describe the theoretical setup for the convergence result.

A standard assumption in convex optimization is the assumption of strong convexity of the function. However, in a game, the operator,

$$\boldsymbol{v} : \boldsymbol{\omega} \mapsto \left[\nabla_{\boldsymbol{\theta}} \mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\omega}),\ \nabla_{\boldsymbol{\varphi}} \mathcal{L}^{(\boldsymbol{\varphi})}(\boldsymbol{\omega})\right]^\top, \tag{8}$$

associated with the updates is not anymore the gradient of a single function. In order to make a similar assumption in the context of games, the optimization literature considers the notion of *strong monotonicity*.

Definition 1. *$F : \boldsymbol{\omega} \mapsto (F_1(\boldsymbol{\omega}), F_2(\boldsymbol{\omega})) \in \mathbb{R}^{d+p}$ is said to be $(\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}})$-strongly monotone if $\forall\, \boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega \subset \mathbb{R}^{p+d}$,*

$$(F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') \geq \mu_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|^2 + \mu_{\boldsymbol{\varphi}} \|\boldsymbol{\varphi} - \boldsymbol{\varphi}'\|^2 .$$

where we write $\boldsymbol{\omega} := (\boldsymbol{\theta}, \boldsymbol{\varphi}) \in \mathbb{R}^{d+p}$. A monotone operator *is a $(0, 0)$-strongly monotone operator.*

This definition is a generalization of strong convexity for operators: if f is μ-strongly convex, then ∇f is a μ-monotone operator. Note that, in this definition, we used the same *weighted Euclidean norm*

$$\Omega(\boldsymbol{\theta}, \boldsymbol{\varphi}) := \mu_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|^2 + \mu_{\boldsymbol{\varphi}} \|\boldsymbol{\varphi} - \boldsymbol{\varphi}'\|^2 , \tag{9}$$

as Palaniappan and Bach (2016). They point out that this rescaling of the Euclidean norm is crucial in order to balance the respective players' objective and getting better constants in the convergence result.

Assumption 1. *For $1 \leq i \leq n$, the functions $\mathcal{L}_i^{(\boldsymbol{\theta})}$ and $\mathcal{L}_i^{(\boldsymbol{\varphi})}$ are respectively $L_i^{(\boldsymbol{\theta})}$ and $L_i^{(\boldsymbol{\varphi})}$-smooth and the associated game operator (8) is $(\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}})$-strongly monotone.*

A function is said to be L-smooth if its gradient is L-Lipschitz. Under this smoothness assumption on each cost function of the game operator, we can define a smoothness constant adapted to the non-uniform sampling scheme of our stochastic algorithm:

$$\bar{L}(\pi)^2 := \frac{1}{n} \sum_{i=1}^{n} \frac{1}{n \pi_i} L_i^2 . \tag{10}$$

The standard *uniform sampling scheme* corresponds to $\pi_i := \frac{1}{n}$ and the optimal *non-uniform* sampling scheme corresponds to $\tilde{\pi}_i := \frac{L_i}{\sum_{i=1}^n L_i}$. We always have the bounds:

$$\bar{L}(\tilde{\pi})^2 = \Big(\frac{1}{n} \sum_{i=1}^{n} L_i\Big)^2 \leq \bar{L}(\pi)^2 = \frac{1}{n} \sum_{i=1}^{n} L_i^2 . \tag{11}$$

We now present our convergence result for SVRE with non-uniform sampling (to make our constants more comparable to those of Palaniappan and Bach (2016)), but note that we have used uniform sampling in all our experiments (for simplicity).

Theorem 3. *Under Assumption 1, after t iterations, the iterate $\boldsymbol{\omega}_t := (\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)$ computed by SVRE (Alg. 1) with step-size $\eta_{\boldsymbol{\theta}} = \frac{1}{12 \bar{L}_{\boldsymbol{\theta}}}$ and $\eta_{\boldsymbol{\varphi}} = \frac{1}{12 \bar{L}_{\boldsymbol{\varphi}}}$ and sampling scheme $(\pi_{\boldsymbol{\theta}}, \pi_{\boldsymbol{\varphi}})$ verifies:*

$$\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] \leq \Big(1 - \frac{1}{12} \min\Big\{\frac{\mu_{\boldsymbol{\theta}}}{\bar{L}_{\boldsymbol{\theta}}}, \frac{\mu_{\boldsymbol{\varphi}}}{\bar{L}_{\boldsymbol{\varphi}}}, \frac{1}{n}\Big\}\Big)^t \mathbb{E}[\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}^*\|_2^2],$$

where $\bar{L}_{\boldsymbol{\theta}}(\pi_{\boldsymbol{\theta}})$ and $\bar{L}_{\boldsymbol{\varphi}}(\pi_{\boldsymbol{\varphi}})$ are defined in (10).

We prove this theorem in § B.2. We can notice that only the respective *condition numbers* of $\mathcal{L}^{(\boldsymbol{\theta})}$ and $\mathcal{L}^{(\boldsymbol{\varphi})}$ defined as $\kappa_{\boldsymbol{\theta}} := \frac{\mu_{\boldsymbol{\theta}}}{\bar{L}_{\boldsymbol{\theta}}}$ and $\kappa_{\boldsymbol{\varphi}} := \frac{\mu_{\boldsymbol{\varphi}}}{\bar{L}_{\boldsymbol{\varphi}}}$ appear in our convergence rate. The convergence rate of the (non-accelerated) algorithm of Palaniappan and Bach (2016) depends on the product of these condition numbers $\kappa_{\boldsymbol{\varphi}} \kappa_{\boldsymbol{\theta}}$. They avoid a dependence on the maximum of the condition numbers squared $\max\{\kappa_{\boldsymbol{\varphi}}^2, \kappa_{\boldsymbol{\theta}}^2\}$ by using the weighted Euclidean norm $\Omega(\boldsymbol{\theta}, \boldsymbol{\varphi})$ defined in (9) and rescaling the functions $\mathcal{L}^{(\boldsymbol{\theta})}$ and $\mathcal{L}^{(\boldsymbol{\varphi})}$ with their strong-monotonicity constant. However, this rescaling trick suffers from two issues: (i) We do not know in practice a good estimate of the strong monotonicity constant, which was *not* the case in Palaniappan and Bach (2016)'s application. (ii) The algorithm does not adapt to (larger) local strong-monotonicity. This property is fundamental in non-convex optimization since we want the algorithm to exploit the (potential) local stability properties of a stationary point.

4.3. Motivating example

The example presented in (2) seems to be a challenging problem in the stochastic setting since even the stochastic extragradient method fails to find the Nash equilibrium

Algorithm 1 Pseudocode for SVRE.

Input: Stopping time T, learning rates $\eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{\varphi}}$, both players' losses $\mathcal{L}^{(\boldsymbol{\theta})}$ and $\mathcal{L}^{(\boldsymbol{\varphi})}$.
Initialize: $\boldsymbol{\theta}_0$, $\boldsymbol{\varphi}_0$
for $t = 0$ **to** $T-1$ **do**
  $\boldsymbol{\varphi}^{\mathcal{S}} = \boldsymbol{\varphi}_t$ and $\boldsymbol{\mu}_{\boldsymbol{\varphi}}^{\mathcal{S}} = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\varphi}} \mathcal{L}_i^{(\boldsymbol{\varphi})}(\boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$
  $\boldsymbol{\theta}^{\mathcal{S}} = \boldsymbol{\theta}_t$ and $\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\mathcal{S}} = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i^{(\boldsymbol{\theta})}(\boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$
  $N \sim \text{Geom}\left(1/n\right)$ *(Sample length of the epoch)*
  for $i = 0$ **to** $N-1$ **do** {*Beginning of the epoch*}
    Sample $i_{\boldsymbol{\theta}} \sim \pi_{\boldsymbol{\theta}}$, $i_{\boldsymbol{\varphi}} \sim \pi_{\boldsymbol{\varphi}}$, do **extrapolation:**
    $\boldsymbol{\varphi}_{t+1/2} = \boldsymbol{\varphi}_t - \eta_{\boldsymbol{\varphi}} \boldsymbol{d}_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$ ▷ (6)
    $\boldsymbol{\theta}_{t+1/2} = \boldsymbol{\theta}_t - \eta_{\boldsymbol{\theta}} \boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$ ▷ (7)
    Sample $i_{\boldsymbol{\theta}} \sim \pi_{\boldsymbol{\theta}}$, $i_{\boldsymbol{\varphi}} \sim \pi_{\boldsymbol{\varphi}}$, do **update:**
    $\boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t - \eta_{\boldsymbol{\varphi}} \boldsymbol{d}_{\boldsymbol{\varphi}}(\boldsymbol{\theta}_{t+\frac{1}{2}}, \boldsymbol{\varphi}_{t+\frac{1}{2}}, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$ ▷ (6)
    $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_{\boldsymbol{\theta}} \boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{\theta}_{t+\frac{1}{2}}, \boldsymbol{\varphi}_{t+\frac{1}{2}}, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$ ▷ (7)
  end for
end for
Output: $\boldsymbol{\theta}$, $\boldsymbol{\varphi}$

of this game. This problem is actually an *unconstrained* version of (Gidel et al., 2019, Fig. 3). To explore how SVRE behaves on this problem, we decided to test its empirical performance. We set $n = d = 100$, and drew $[\boldsymbol{A}_i]_{kl} = \delta_{kli}$ 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. Our optimization problem was:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \max_{\boldsymbol{\varphi}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{\theta}^\top \boldsymbol{b}_i + \boldsymbol{\theta}^\top \boldsymbol{A}_i \boldsymbol{\varphi} + \boldsymbol{c}_i^\top \boldsymbol{\varphi}). \tag{12}$$

We compared the following algorithms (which are all using uniform sampling): (i) AltSGD: Standard stochastic gradient method alternating the update of each player. This is the standard method to train GANs. (ii) SVRE: The algorithm presented in this paper Alg. 1. The Avg prefix correspond to the *uniform average* of the iterates, $\bar{\omega} := \frac{1}{t}\sum_{s=0}^{t-1} \boldsymbol{\omega}_s$. We observe in Fig. 2 that AvgSVRE converges sublinearly (whereas AvgAltSGD fails to converge). This motivates a new variant of SVRE that is based on the ideas developed in § 3.2, that even if the averaged iterate converge we do not compute the gradient at that point and thus we do not benefit from the fact that this iterate is closer to the optimum. Thus the idea is to occasionally restart the algorithm, i.e., consider the averaged iterate as the new starting point of our algorithm and compute the gradient at that point. Restart goes well with SVRG since we already occasionally stop the inner loop to recompute a new snapshot, we make use of this pause to decide (with a probability p to be fixed) whether or not we restart our algorithm taking the snapshot at point $\bar{\boldsymbol{\omega}}_t$ instead of $\boldsymbol{\omega}_t$. This variant of SVRG is described in Alg. 3 in § D and the variant combining VRAd is described in § C.1.

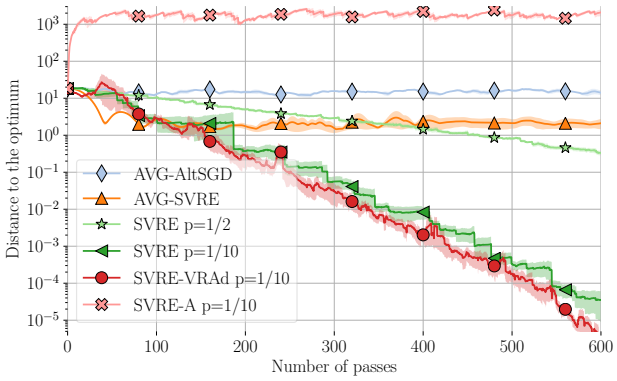


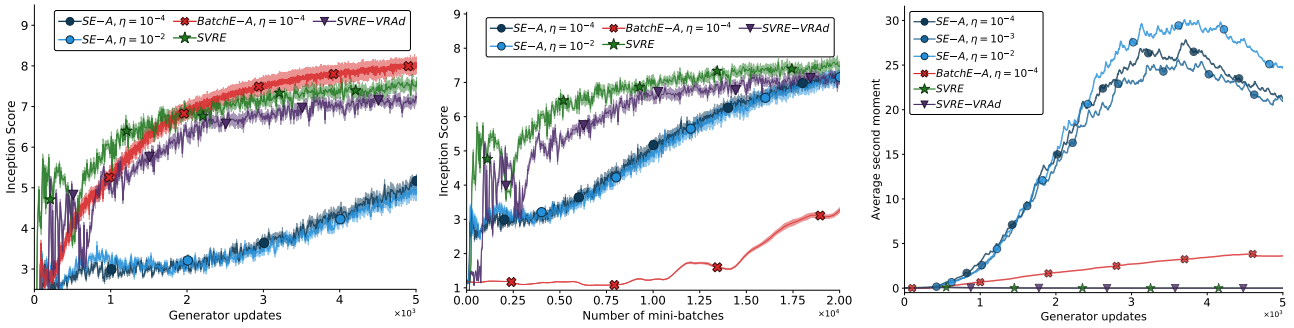
Figure 2: Distance to the Nash equilibrium (NE) of (12). We ran the experiment with 5 different seeds and plotted the average distances to the NE and their standard deviation.

In Fig. 2 we observe that the only method that converges is SVRE and its variants. Note that we do not provide any convergence guarantees for Alg. 3 and leave its analysis for future work. However, it seems interesting that, to our knowledge, this algorithm is the only stochastic algorithm (excluding batch extragradient since it is not a stochastic algorithm) that converge for (2). Note that we tried all the algorithms presented in Fig. 3 from Gidel et al. (2019) on this *unconstrained* problem and that all of them diverge.

5. Experiments

Datasets. To evaluate SVRE, we used the following datasets: (i) **MNIST** (Lecun and Cortes), (ii) **CIFAR-10** (Krizhevsky, 2009, §3), (iii) **SVHN** (Netzer et al., 2011), as well as (iv) **ImageNet** ILSVRC 2012 (Russakovsky et al., 2015), using 28×28, $3 \times 32 \times 32$, $3 \times 32 \times 32$, and $3 \times 64 \times 64$ resolution, respectively.

Metrics. To evaluate the performance of SVRE, we conducted experiments on image synthesis with GAN and used the **Inception score** (IS, Salimans et al., 2016) and the **Fréchet Inception distance** (FID, Heusel et al., 2017) as performance metrics. IS takes into account both the sample "diversity" and how realistic the samples look, by considering the class distribution and prediction confidence of a trained Inception network (Szegedy et al., 2015), respectively. On the other hand, FID compares the synthetic images with those of the training dataset, by embedding large samples of the two in a lower dimensional space, using trained Inception network. To gain insights if SVRE indeed reduces the variance of the gradient estimates (over the iterations) we used the **second moment estimate** (uncentered variance, herein denoted as *SME*), computed with an exponentially moving average. SME is computed for each parameter as done in Adam (Kingma and Ba, 2015), and finally, we plot the averaged value over all the parameters.



(a) IS (higher is better) (b) IS (higher is better) (c) Generator

Figure 3: Stochastic, full-batch and variance reduced extragradient optimization on **MNIST** (see § 5 for naming of methods). We used $\eta = 10^{-2}$ for the SVRE based methods. The input space is $1\times28\times28$, see § C for details on the implementation. *SE-A*, $\eta = 10^{-3}$ achieves similar IS performances as $\eta = 10^{-2}$ & $\eta = 10^{-4}$, omitted from 3a & 3b.

We refer the reader to § E for details on these metrics.

Optimization methods. We conduct experiments using the following optimization methods for GANs: (i) **BatchE:** full–batch extragradient, (ii) **SE:** stochastic extragradient, and (iii) **SVRE:** stochastic variance reduced extragradient. These can be combined with adaptive learning rate methods such as *Adam*, or with parameter averaging and warm-start of it, hereafter denoted as **–A**, **AVG–** and **WA–**, respectively. With **BatchE–A** and **SE–A** we denote respectively full–batch and stochastic extragradient that use Adam. In § C.1, we present a variant of Adam adapted to variance reduced algorithms, that is referred as **VRAd**. All our experiments use mini-batching with uniform probabilities; the mini-batch version of SVRE for GANs is described in more details in § C.2.

DNN architectures. For experiments on **MNIST**, we use the DCGAN (Radford et al., 2016) architecture. For the remaining datasets, we used the SAGAN (Zhang et al., 2018) architecture, as it is used by Brock et al. (2019) demonstrating state-of-art performances on **ImageNet**. The focus of our experiments is to compare SVRE with baselines in a normalized and realistic setting. However, we lack the needed computational resources to do the extensive hyperparameter tuning typically required to obtain new state-of-the-art results with the latest architectures. In particular, for efficiency, we used architectues of approximately half of the depth of the CIFAR-10 architectures listed in Miyato et al. (2018) (see § E for the details on our implementation).

5.1. Results

We conducted experiments on the **MNIST** common benchmark, for which full-batch extragradient is feasible to compute. Fig. 3 depicts the **IS** metrics while using either a stochastic, full-batch or variance reduced version of extragradient (see details of SVRE-GAN in § C.2). For the stochastic baseline (SE), we always use the *Adam* optimization method, as originally proposed (Gidel et al., 2019). From Fig. 3a, we observe that in terms of the number of parameter updates, SVRE performs similarly to BatchE-A.

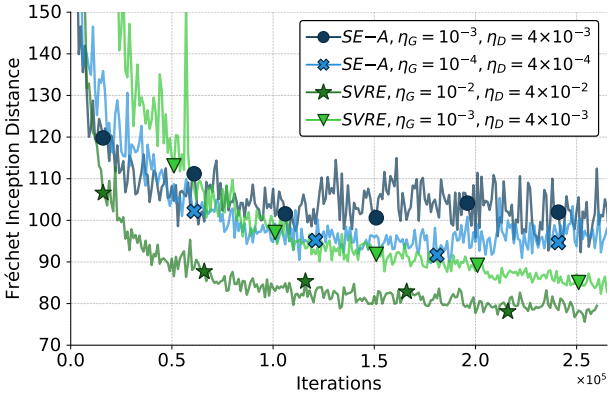


Figure 4: FID comparison (lower is better) between *SVRE* and the *SE–A* baseline on **Imagenet**.

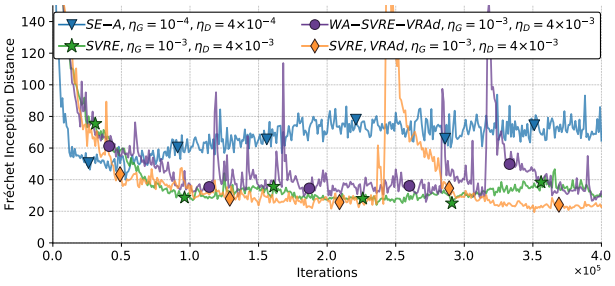


Figure 5: Comparison of the *SVRE* methods with the *SE* baseline on **SVHN** using the FID metric, where lower is better. *SE-A*, $\eta = 10^{-3}$ did not converge and is omitted for clearer illustration.

	Inception Score			Fréchet Inception Distance		
	SE–A	SVRE	SVRE–V	SE–A	SVRE	SVRE–V
MNIST	8.62	8.58	8.56	0.17	0.15	0.18
CIFAR-10	6.61	6.50	6.67	37.20	39.20	38.88
SVHN	2.83	3.01	**3.04**	39.95	24.01	**19.40**
ImageNet	7.22	**8.08**	7.50	89.40	**75.60**	81.24

Table 2: Summary: best obtained IS and FID scores for the different optimization methods, for a fixed number of iterations (see § E), where *SVRE–V*, denotes *SVRE–VRAd* (see § 5). The standard deviation of the Inception scores is around 0.1 and is omitted. Although the IS metric does not make much sense on **SVHN** due to the dataset properties (see § E.1), we include it for completeness.

Note, however, that the latter requires significantly more computation: Fig. 3b depicts the IS metric using the number of mini-batch computations as x-axis (a surrogate for the wall-clock time, see discussion below). We observe that, as SE-A has slower per-iteration convergence rate, SVRE converges faster on this set of experiments.

Computational cost. The relative cost of one pass over the dataset for SVRE versus vanilla SGD is a factor of 5: the full batch gradient is computed (on average) after one pass over the dataset, giving a slowdown of 2; the factor 5 takes into account the extra stochastic gradient computations for the variance reduction, as well as the extrapolation step overhead. Note that some training methods for GANs have similar overhead, e.g. how Arjovsky et al. (2017) used 5 discriminator updates per generator update for training WGAN. Moreover, as SVRE provides less noisy gradient, it may converge faster per iteration, compensating the extra per-update cost. In Fig. 3b, the x-axis uses an implementation-independent surrogate for wall-clock time that counts the number of mini-batch gradient computations, which is the bottleneck step.

Second moment estimate and Adam. Fig. 3c depicts the averaged second-moment estimate for parameters of the Generator, where we observe that SVRE effectively reduces it over the iterations. The reduction of these values may be the reason why Adam combined with SVRE performs poorly (as these values appear in the denominator of Adam, see § C.1). Nonetheless, SVRE produces good results on several high dimensional datasets. To our knowledge, SVRE is the first optimization method with a constant step size that has worked empirically for GANs on non-trivial datasets.

Comparison on larger datasets. In Fig. 4 & 5, we compare SVRE with the SE-A baseline on the **SVHN** and **ImageNet** datasets. We observe that although SE–A in some experiments obtains better performances in the early iterations (starts to converge faster), SVRE allows for obtaining better final performances. In Tab. 2 we summarize the best scores obtained on **MNIST**, **CIFAR-10**, **SVHN** and **ImageNet**. Note that our experimental part was devised to investigate if SVRE has potential advantages in a realistic GAN setup, and *we did not perform exhaustive hyperparameter search and multiple runs with different random seeds, due to a lack of computational resources.* § F lists additional experimental analyses.

Comparison using deeper architectures on CIFAR-10 and SVHN. We report here our preliminary results using the ResNet (He et al., 2015) architecture (see § E.2) used for **CIFAR-10** and **SVHN** in (Miyato et al., 2018), on these two datasets as a reference. We compare SE–A and SVRE after 800K iterations of training, and we obtain FID scores of: (i) SVHN: 5.14 and 6.85 for SE–A and SVRE, resp.; (ii) CIFAR-10: 18.65, 23.56 for SE–A and SVRE, resp. (see § F for details). We observe that these experiments are less stable, and a fair comparison requires more exhaustive hyper-parameter search. More precisely, SE–A diverged in 100% of the experiments, while using ratio of $1:5$ of updates of G and D, resp. improved the stability. On the other hand, SVRE *did not diverge in any of the experiments*, but on deeper architectures we observe that SVRE takes longer time to converge compared to SE–A. Moreover, warm-starting from the baseline of 18.65 after 150K iterations and then continuing by using SVRE, we obtain a FID of 17.72. This could indicate that SVRE may largely benefit if combined with adaptive step size method, what we leave for future work.

6. Related work

Nowadays, two main stochastic variance reduction algorithm considered in the literature are SAGA (Defazio et al., 2014) and SVRG (Johnson and Zhang, 2013). The former is memory based, storing the latest stochastic gradient seen for each sample whereas the latter is epoch based, computing a full batch gradient at the beginning of each epoch. Interestingly, Hofmann et al. (2015) provides a randomized version (for the epoch length) of SVRG under a framework unifying SVRG and SAGA, and this is under that framework that we provide our analysis, thus proving at the same time for SVRE and SAGA with extragradient.

Surprisingly, there exist only a few works on variance reduction methods for monotone operators: (Palaniappan and Bach, 2016) and (Davis, 2016). As mentioned by Palaniappan and Bach (2016), the latter requires a co-coercivity assumption on the operator and thus only convex optimization is considered. Our work provides a new way to use variance reduction for monotone operators, using the ex-

tragradient method (Korpelevich, 1976). Recently, Iusem et al. (2017) proposed an extragradient method with variance reduction for an *infinite sum* of operators. They use mini-batches of growing size in order to reduce the variance of their algorithm and to converge with a constant step-size. However, this growing mini-batch size is prohibitively expensive in our application. Moreover, they are not using the SAGA/SVRG style of updates exploiting the finite sum formulation, leading to sublinear convergence rate, while our method benefits from a linear convergence rate exploiting the finite sum assumption.

Daskalakis et al. (2018) proposed a method called Optimistic-Adam inspired by game theory. This method is closely related to extragradient, with slightly different update scheme. More recently, Gidel et al. (2019) proposed to use extragradient to train GANs, introducing a method called ExtraAdam. This method outperformed Optimistic-Adam when trained on CIFAR-10. Our work is also an attempt to find principled ways to train GANs. Considering that the game aspect is better handled by the extragradient method, we focus on the optimization issues arising from the noise in the training procedure, an under-considered potential issue in GAN training.

Noise in Deep Learning and Games. In the context of deep learning, despite some very interesting theoretical results on non-convex minimization (Reddi et al., 2016; Allen-Zhu and Hazan, 2016), the effectiveness of variance reduced methods is still an open question, and a recent technical report by Defazio and Bottou (2018) provides negative empirical results on the variance reduction aspect.

In addition, a recent large scale study (Shallue et al., 2018) tried to increase the batch size is a simple way and observed a marginal impact on single objective training, while in contrast it had an effect on GAN results (Brock et al., 2019). In our work we are able to show positive results for variance reduction in a real world deep learning setting. This surprising difference seems to confirm the remarkable discrepancy, that remain poorly understood, between multi-objective optimization and standard minimization.

7. Conclusion

We considered a bilinear game optimization and despite its simplicity compared to real-world GAN optimization problems, we show that stochasticity breaks its convergence. We proposed the stochastic variance reduced extragradient method that combines SVRG with the extragradient method for optimizing games. On the theory side, SVRE improves upon the previous best results for strongly-convex games, whereas empirically, SVRE is the only method that converges for the bilinear game.

Our results showed that SVRE obtained empirically similar convergence speed to Batch-Extragradient on MNIST, with the latter being computationally infeasible for larger datasets. For standard size convolutional neural networks, SVRE showed improved final performances in 2 out of 4 datasets.

Our preliminary experiments with deeper architectures show that SVRE is notably more stable in terms of the choice of hyperparameters, as its stochastic counterpart diverged in all of our experiments, whereas SVRE did not. However, we observe that SVRE takes more iterations to converge when using deeper architectures, suggesting that SVRE can be further extended with adaptive step size method. Considering the notable instabilities when using such architectures and standard optimization methods, our results suggest variance reduction methods as necessary for improving GAN training, unlike single objective optimization.

Acknowledgements

This research was partially supported by the Canada CIFAR AI Chair Program, the Canada Excellence Research Chair in "Data Science for Realtime Decision-making", by the NSERC Discovery Grant RGPIN-2017-06936 and by a Google Focused Research award. Authors would like to thank Compute Canada for providing the GPUs used for this research.

References

Z. Allen-Zhu and E. Hazan. Variance reduction for faster non-convex optimization. In *ICML*, 2016.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

C. Daskalakis, A. Ilyas, V. Syrgkanis, and H. Zeng. Training GANs with optimism. In *ICLR*, 2018.

D. Davis. Smart: The stochastic monotone aggregated root-finding algorithm. *arXiv:1601.00698*, 2016.

A. Defazio and L. Bottou. On the ineffectiveness of variance reduced optimization for deep learning. *arXiv:1812.04529*, 2018.

A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, 2014.

G. Gidel, H. Berard, P. Vincent, and S. Lacoste-Julien. A variational inequality perspective on generative adversarial nets. In *ICLR (to appear)*, 2019.

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

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

P. T. Harker and J.-S. Pang. Finite-dimensional variational inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications. *Mathematical programming*, 1990.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

T. Hofmann, A. Lucchi, S. Lacoste-Julien, and B. McWilliams. Variance reduced stochastic gradient descent with neighbors. In *NIPS*, 2015.

A. Iusem, A. Jofré, R. I. Oliveira, and P. Thompson. Extragradient method with variance reduction for stochastic variational inequalities. *SIAM Journal on Optimization*, 2017.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, 2013.

A. Juditsky, A. Nemirovski, and C. Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 2011.

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

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

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

R. Leblond, F. Pederegosa, and S. Lacoste-Julien. Improved asynchronous parallel optimization analysis for stochastic incremental methods. *JMLR*, 19(81):1–68, 2018.

Y. Lecun and C. Cortes. The MNIST database of handwritten digits. URL http://yann.lecun.com/exdb/mnist/.

J. Li, A. Madry, J. Peebles, and L. Schmidt. On the limitations of first order approximation in GAN dynamics. In *ICML*, 2018.

J. H. Lim and J. C. Ye. Geometric GAN. *arXiv:1705.02894*, 2017.

L. Mescheder, S. Nowozin, and A. Geiger. The numerics of GANs. In *NIPS*, 2017.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011. URL http://ufldl.stanford.edu/housenumbers/.

B. Palaniappan and F. Bach. Stochastic variance reduction methods for saddle-point problems. In *NIPS*, 2016.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola. Stochastic variance reduction for nonconvex optimization. In *ICML*, 2016.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NIPS*, 2016.

T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. In *ICML*, 2013.

M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 2017.

C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl. Measuring the effects of data parallelism on neural network training. *arXiv:1811.03600*, 2018.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *arXiv:1512.00567*, 2015.

A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In *NIPS*, 2017.

L. Xiao and T. Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-Attention Generative Adversarial Networks. *arXiv:1805.08318*, 2018.

A. Definitions and Lemmas

A.1. Smoothness and Monotonicity of the operator

Another important property used is the Lipschitzness of an operator.

Definition 2. *A mapping $F : \mathbb{R}^p \to \mathbb{R}^d$ is said to be L-Lipschitz if,*

$$\|F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}')\|_2 \leq L\|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2\,, \quad \forall \boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega\,. \tag{13}$$

Definition 3. *A differentiable function $f : \Omega \to \mathbb{R}$ is said to be* μ-strongly convex *if*

$$f(\boldsymbol{\omega}) \geq f(\boldsymbol{\omega}') + \nabla f(\boldsymbol{\omega}')^\top(\boldsymbol{\omega} - \boldsymbol{\omega}') + \frac{\mu}{2}\|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 \quad \forall \boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega\,. \tag{14}$$

Definition 4. *A function $(\boldsymbol{\theta}, \boldsymbol{\varphi}) \mapsto \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi})$ is said convex-concave if $\mathcal{L}(\cdot, \boldsymbol{\varphi})$ is convex for all $\boldsymbol{\varphi} \in \Phi$ and $\mathcal{L}(\boldsymbol{\theta}, \cdot)$ is concave for all $\boldsymbol{\theta} \in \Theta$. An $\mathcal{L}$ is said to be μ-strongly convex concave if $(\boldsymbol{\theta}, \boldsymbol{\varphi}) \mapsto \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) - \frac{\mu}{2}\|\boldsymbol{\theta}\|_2^2 + \frac{\mu}{2}\|\boldsymbol{\varphi}\|_2^2$ is convex concave.*

Definition 5. *For $\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}} > 0$, an operator $F : \boldsymbol{\omega} \mapsto (F_1(\boldsymbol{\omega}), F_2(\boldsymbol{\omega})) \in \mathbb{R}^{d+p}$ is said to be $(\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}})$-strongly monotone if $\forall\, \boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega \subset \mathbb{R}^{p+d}$,*

$$(F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}'))^\top(\boldsymbol{\omega} - \boldsymbol{\omega}') \geq \mu_{\boldsymbol{\theta}}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|^2 + \mu_{\boldsymbol{\varphi}}\|\boldsymbol{\varphi} - \boldsymbol{\varphi}'\|^2\,.$$

where we noted $\boldsymbol{\omega} := (\boldsymbol{\theta}, \boldsymbol{\varphi}) \in \mathbb{R}^{d+p}$.

B. Proof of Theorems

B.1. Proof of Theorem 2

Proof. Let us consider the optimization problem,

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \max_{\boldsymbol{\varphi} \in \mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\theta}^\top \boldsymbol{A}_i \boldsymbol{\varphi}\,, \tag{15}$$

where $[\boldsymbol{A}_i]_{kl} = 1$ if $k = l = i$ and 0 otherwise. Note that $(\boldsymbol{A}_i)^\top = \boldsymbol{A}_i$ for $1 \leq i \leq n$. Let us consider the extragradient method where to compute an unbiased estimator of the gradients at $(\boldsymbol{\theta}, \boldsymbol{\varphi})$ we sample $i \in \{1, \dots, n\}$ and use $[\boldsymbol{A}_i\boldsymbol{\theta},\ \boldsymbol{A}_i\boldsymbol{\varphi}]$ as estimator of the vector flow.

In this proof we note, $\boldsymbol{A}_I := \sum_{i \in I} \boldsymbol{A}_i$ and $\boldsymbol{\theta}^{(I)}$ the vector such that $[\boldsymbol{\theta}^{(I)}]_i = [\boldsymbol{\theta}]_i$ if $i \in I$ and 0 otherwise. Note that $\boldsymbol{A}_I\boldsymbol{\theta} = \boldsymbol{\theta}^{(I)}$ and that $\boldsymbol{A}_I\boldsymbol{A}_J = \boldsymbol{A}_{I \cap J}$.

Thus the extragradient update rule can be noted as

$$\begin{cases} \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{A}_I(\boldsymbol{\varphi}_t + \eta \boldsymbol{A}_J\boldsymbol{\theta}_t) \\ \boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t + \eta \boldsymbol{A}_I(\boldsymbol{\theta}_t - \eta \boldsymbol{A}_J\boldsymbol{\varphi}_t)\,, \end{cases} \tag{16}$$

where I is the mini-batch (without replacement) sampled for the update and J the mini-batch (without replacement) sampled for the extrapolation.

We can thus notice that, when $I \cap J = \emptyset$, we have

$$\begin{cases} \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{\varphi}_t^{(I)} \\ \boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t + \eta \boldsymbol{\theta}_t^{(I)}\,, \end{cases} \tag{17}$$

and otherwise,

$$\begin{cases} \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{\varphi}_t^{(I)} - \eta^2 \boldsymbol{\theta}_t^{(I \cap J)} \\ \boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t + \eta \boldsymbol{\theta}_t^{(I)} - \eta^2 \boldsymbol{\varphi}_t^{(I \cap J)}\,. \end{cases} \tag{18}$$

The intuition is that, on one hand, when $I \cap J = \emptyset$ (which happens with high probability when $|I| << n$, e.g., when $|I| = 1$, $\mathbb{P}(I \cap J = \emptyset) = 1 - 1/n$), the algorithm performs an update that get away from the Nash equilibrium:

$$(17) \;\Rightarrow\; N_{t+1} = N_t + \eta^2 N_t^{(I)}\,, \tag{19}$$

where $N_t := \|\boldsymbol{\theta}_t\|^2 + \|\boldsymbol{\varphi}_t\|^2$ and $N_t^{(I)} := \|\boldsymbol{\theta}_t^{(I)}\|^2 + \|\boldsymbol{\varphi}_t^{(I)}\|^2$. On the other hand, The "good" updates only happen when $I \cap J$ is large (which happen with low probability, e.g., when $|I| = 1$, $\mathbb{P}(I \cap J \neq \emptyset) = 1/n$):

$$(18) \;\Rightarrow\; N_{t+1} = N_t - \eta^2(2N_t^{(I\cap J)} - N_t^{(I)}) + \eta^4 N_t^{(I\cap J)}\,. \tag{20}$$

Conditioning on $\boldsymbol{\theta}_t$ and $\boldsymbol{\varphi}_t$, we get that

$$\mathbb{E}[N_t^{(I\cap J)}|\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t] = \sum_{i=1}^n \mathbb{P}(i \in I \cap J)([\boldsymbol{\theta}_t]_i^2 + [\boldsymbol{\varphi}_t]_i^2) \quad \text{and} \quad \mathbb{P}(i \in I \cap J) = \mathbb{P}(i \in I)\mathbb{P}(i \in J) = \frac{|I|^2}{n^2}\,. \tag{21}$$

Leading to,

$$\mathbb{E}[N_t^{(I\cap J)}|\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t] = \frac{|I|^2}{n^2}\sum_{i=1}^n([\boldsymbol{\theta}_t]_i^2 + [\boldsymbol{\varphi}_t]_i^2) = \frac{|I|^2}{n^2}N_t \quad \text{and} \quad \mathbb{E}[N_t^{(I)}|\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t] = \frac{|I|}{n}N_t\,. \tag{22}$$

Plugging these expectations in (20), we get that,

$$\mathbb{E}[N_{t+1}|\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t] = N_t\left(1 - 2\eta^2\frac{|I|^2}{n^2} + \eta^2\frac{|I|}{n} + \frac{|I|^2}{n^2}\eta^4\right)\,. \tag{23}$$

Consequently,

$$\mathbb{E}[N_{t+1}] = \left(1 - 2\eta^2\frac{|I|^2}{n^2} + \eta^2\frac{|I|}{n} + \frac{|I|^2}{n^2}\eta^4\right)\mathbb{E}[N_t]\,. \tag{24}$$

Thus, N_t converge if and only if $\exists \eta > 0\,,\; s.t.\,,\; 2|I| > n + |I|\eta^2$ which is not possible when $2|I| \leq n$.

To sum-up, if $|I|$ is not large enough (more precisely if $2|I| \leq n$), we have the geometric divergence of the quantity $\mathbb{E}[N_t] := \mathbb{E}[\|\boldsymbol{\theta}_t\|^2 + \|\boldsymbol{\varphi}_t\|^2]$. □

B.2. Proof of Theorem 3

Setting of the Proof. We will prove a slightly more general result than Theorem 3. We will work in the context of monotone operator. Let us consider the *general* extrapolation update rule,

$$\begin{cases} \text{Extrapolation:} & \boldsymbol{\omega}_{t+\frac{1}{2}} = P_{\mathcal{X}}[\boldsymbol{\omega}_t - \eta_t \boldsymbol{g}_t] \\ \text{Update:} & \boldsymbol{\omega}_{t+1} = P_{\mathcal{X}}[\boldsymbol{\omega}_t - \eta_t \boldsymbol{g}_{t+1/2}]\,. \end{cases} \tag{25}$$

where $\boldsymbol{g}_t$ depends on $\boldsymbol{\omega}_t$ and $\boldsymbol{g}_{t+1/2}$ depends on $\boldsymbol{\omega}_{t+1/2}$. Here $P_{\mathcal{X}}$ denotes the Euclidean projection operator on $\mathcal{X}$, i.e. $P_{\mathcal{X}}[\boldsymbol{x}] := \arg\min_{\boldsymbol{x}' \in \mathcal{X}} \|\boldsymbol{x}' - \boldsymbol{x}\|_2^2$.

This update rule generalizes (EG) for 2-player games (2P-G) and ExtraSVRG (Alg. 2).

Let us first state a lemma standard in convex analysis (see for instance (Boyd and Vandenberghe, 2004)),

Lemma 1. *Let $\boldsymbol{\omega} \in \Omega$ and $\boldsymbol{\omega}^+ := P_\Omega(\boldsymbol{\omega} + \boldsymbol{u})$ then for all $\boldsymbol{\omega}' \in \Omega$ we have,*

$$\|\boldsymbol{\omega}^+ - \boldsymbol{\omega}'\|_2^2 \leq \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 + 2\boldsymbol{u}^\top(\boldsymbol{\omega}^+ - \boldsymbol{\omega}') - \|\boldsymbol{\omega}^+ - \boldsymbol{\omega}\|_2^2\,. \tag{26}$$

Proof of Lemma 1. We start by simply developing,

$$\begin{aligned}\|\boldsymbol{\omega}^+ - \boldsymbol{\omega}'\|_2^2 = \|(\boldsymbol{\omega}^+ - \boldsymbol{\omega}) + (\boldsymbol{\omega} - \boldsymbol{\omega}')\|_2^2 &= \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 + 2(\boldsymbol{\omega}^+ - \boldsymbol{\omega})^\top(\boldsymbol{\omega} - \boldsymbol{\omega}') + \|\boldsymbol{\omega}^+ - \boldsymbol{\omega}\|_2^2 \\ &= \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 + 2(\boldsymbol{\omega}^+ - \boldsymbol{\omega})^\top(\boldsymbol{\omega}^+ - \boldsymbol{\omega}') - \|\boldsymbol{\omega}^+ - \boldsymbol{\omega}\|_2^2\,.\end{aligned}$$

Then since $\boldsymbol{\omega}^+$ is the projection onto the convex set Ω of $\boldsymbol{\omega} + \boldsymbol{u}$ we have that $(\boldsymbol{\omega}^+ - (\boldsymbol{\omega} + \boldsymbol{u}))^\top(\boldsymbol{\omega}^+ - \boldsymbol{\omega}') \leq 0\,,\; \forall\, \boldsymbol{\omega}' \in \Omega$, leading to the result of the Lemma. □

Lemma 2. *If F is $(\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}})$-strongly monotone for any $\boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega$ we have,*

$$\mu_{\boldsymbol{\theta}}\left(\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2^2 - 2\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|_2^2\right) + \mu_{\boldsymbol{\varphi}}\left(\|\boldsymbol{\varphi} - \boldsymbol{\varphi}^*\|_2^2 - 2\|\boldsymbol{\varphi}' - \boldsymbol{\varphi}\|_2^2\right) \leq 2F(\boldsymbol{\omega}')^\top(\boldsymbol{\omega}' - \boldsymbol{\omega}^*)\,, \quad \forall\, \boldsymbol{\omega}^* \in \Omega^*\,, \tag{27}$$

where we noted $\boldsymbol{\omega} := (\boldsymbol{\theta}, \boldsymbol{\varphi})$.

Proof. By $(\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}})$-strong monotonicity and optimality of $\boldsymbol{\omega}^*$,

$$2\mu_{\boldsymbol{\theta}}\|\boldsymbol{\theta}' - \boldsymbol{\theta}^*\|_2^2 + 2\mu_{\boldsymbol{\varphi}}\|\boldsymbol{\varphi}' - \boldsymbol{\varphi}^*\|_2^2 \leq 2F(\boldsymbol{\omega}^*)^\top(\boldsymbol{\omega}' - \boldsymbol{\omega}^*) + 2\mu_{\boldsymbol{\theta}}\|\boldsymbol{\theta}' - \boldsymbol{\theta}^*\|_2^2 + 2\mu_{\boldsymbol{\varphi}}\|\boldsymbol{\varphi}' - \boldsymbol{\varphi}^*\|_2^2 \tag{28}$$

$$\leq 2F(\boldsymbol{\omega}')^\top(\boldsymbol{\omega}' - \boldsymbol{\omega}^*)\,, \tag{29}$$

and then we use the inequality $2\|\boldsymbol{a}' - \boldsymbol{a}^*\|_2^2 \geq \|\boldsymbol{a} - \boldsymbol{a}^*\|_2^2 - 2\|\boldsymbol{a}' - \boldsymbol{a}\|_2^2$ to get the result claimed. □

Using this update rule we can thus deduce the following lemma, the derivation of this lemma is very similar from the derivation of Harker and Pang (1990, Lemma 12.1.10).

Lemma 3. *Considering the update rule (25), we have for any $\boldsymbol{\omega} \in \Omega$ and any $t \geq 0$,*

$$2\eta_t \boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}) \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 + \eta_t^2\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|_2^2\,. \tag{30}$$

Proof. Applying Lem. 1 for $(\boldsymbol{\omega}, \boldsymbol{u}, \boldsymbol{\omega}^+, \boldsymbol{\omega}') = (\boldsymbol{\omega}_t, -\eta_t\boldsymbol{g}_{t+1/2}, \boldsymbol{\omega}_{t+1}, \boldsymbol{\omega})$ and $(\boldsymbol{\omega}, \boldsymbol{u}, \boldsymbol{\omega}^+, \boldsymbol{\omega}') = (\boldsymbol{\omega}_t, -\eta_t\boldsymbol{g}_t, \boldsymbol{\omega}_{t+1/2}, \boldsymbol{\omega}_{t+1})$, we get,

$$\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t\boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}) - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t\|_2^2\,, \tag{31}$$

and

$$\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_{t+1}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1}\|_2^2 - 2\eta_t\boldsymbol{g}_t^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2\,. \tag{32}$$

Summing (31) and (32) we get,

$$\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t\boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}) \tag{33}$$

$$- 2\eta_t\boldsymbol{g}_t^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_{t+1}\|_2^2 \tag{34}$$

$$= \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t\boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}) - \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_{t+1}\|_2^2$$

$$- 2\eta_t(\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2})^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_{t+1})\,. \tag{35}$$

Then, we can use the Young's inequality $-2a^\top b \leq \|a\|_2^2 + \|b\|_2^2$ to get,

$$\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t\boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}) + \eta_t^2\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|_2^2$$

$$+ \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_{t+1}\|_2^2 - \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_{t+1}\|_2^2 \tag{36}$$

$$= \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t\boldsymbol{g}_{t+1/2}^\top(\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}) + \eta_t^2\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|_2^2 - \|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2\,. \tag{37}$$

□

Note that if we would have set $\boldsymbol{g}_t = \mathbf{0}$ and $\boldsymbol{g}_{t+1/2}$ any estimate of the gradient at $\boldsymbol{\omega}_t$ we recover the standard lemma for gradient method.

Let us consider *unbiased* estimates of the gradient,

$$\boldsymbol{g}_i(\boldsymbol{\omega}) := \frac{1}{n\pi_i}\left(F_{i_k}(\boldsymbol{\omega}) - \boldsymbol{\alpha}_{i_k}\right) + \bar{\boldsymbol{\alpha}}\,, \tag{38}$$

where $\bar{\boldsymbol{\alpha}} := \frac{1}{n}\sum_{j=1}^n \boldsymbol{\alpha}_j$, the index i_k are (potentially) non-uniformly sampled from $\{1, \ldots, n\}$ with replacement according to $\boldsymbol{\pi}$ and $F(\boldsymbol{\omega}) := \frac{1}{n}\sum_{j=1}^n F_i(\boldsymbol{\omega})$. Hence we have that $\mathbb{E}[\boldsymbol{g}_i(\boldsymbol{\omega})] = F(\boldsymbol{\omega})$, where the expectation is taken respect to the index i sampled from $\boldsymbol{\pi}$.

We will consider a class of algorithm called *uniform memorization algorithms* first introduced by (Hofmann et al., 2015). This class of algorithms describes a large subset of variance reduced algorithms taking advantage of the finite sum formulation such as, SAGA (Defazio et al., 2014), SVRG (Johnson and Zhang, 2013) or q-SAGA and $\mathcal{N}$-SAGA (Hofmann et al., 2015). In this work, we will use a more general definition of such algorithm:

Definition 6 (Extension of (Hofmann et al., 2015))**.** *A uniform q-memorization algorithm evolves iterates $(\boldsymbol{\omega}_t)$ according to (25), with $\boldsymbol{g}_t$ defined in (38) and selecting in each iteration t a random index set J_t of memory locations to update according to,*

$$\boldsymbol{\alpha}_j^{(0)} := F_j(\boldsymbol{\omega}_0)\,, \qquad \boldsymbol{\alpha}_j^{(t+1/2)} := \boldsymbol{\alpha}_j^{(t)},\ \forall j \in \{1,\dots,n\} \qquad \textit{and} \qquad \boldsymbol{\alpha}_j^{(t+1)} := \begin{cases} F_j(\boldsymbol{\omega}_t) & \textit{if } j \in J_t \\ \boldsymbol{\alpha}_j^{(t)} & \textit{otherwise.} \end{cases} \tag{39}$$

such that any j has the same probability q/n to be updated, i.e., $P\{j\} = \sum_{J_t, j\in J_t} P(J_t) = q/n\,,\ \forall j \in \{1,\dots,n\}$.

We have the following lemmas,

Lemma 4. *For any $t \geq 0$, if we consider a q-memorization algorithm with iterates evolving according (25), we have,*

$$\mathbb{E}[\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2] \leq \mathbb{E}[\tfrac{6}{n^2\pi_j^2}\|F_i(\boldsymbol{\omega}_t) - \boldsymbol{\alpha}_i^{(t)}\|^2] + 3\bar{L}^2\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2\,. \tag{40}$$

Proof. We use an extended version of Young's inequality: $\|\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c}\|^2 \leq 3\|\boldsymbol{a}\|^2 + 3\|\boldsymbol{b}\|^2 + 3\|\boldsymbol{c}\|^2$,

$$\begin{aligned}\|\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c}\|^2 &= \|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2 + \|\boldsymbol{c}\|^2 + 2\boldsymbol{a}^\top\boldsymbol{b} + 2\boldsymbol{a}^\top\boldsymbol{c} + 2\boldsymbol{b}^\top\boldsymbol{c} \\ &\leq \|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2 + \|\boldsymbol{c}\|^2 + (\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2) + (\|\boldsymbol{a}\|^2 + \|\boldsymbol{c}\|^2) + (\|\boldsymbol{b}\|^2 + \|\boldsymbol{c}\|^2) \\ &= 3\|\boldsymbol{a}\|^2 + 3\|\boldsymbol{b}\|^2 + 3\|\boldsymbol{c}\|^2\,,\end{aligned}$$

where we used that $2\boldsymbol{a}^\top\boldsymbol{b} \leq +\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2$. Then, we combine this inequality with the definition of q-memorization algorithm: $\boldsymbol{g}_t = \frac{1}{\pi_i}(F_i(\boldsymbol{\omega}_t) - \boldsymbol{\alpha}_i^{(t)}) + \bar{\boldsymbol{\alpha}}^{(t)}$ and $\boldsymbol{g}_{t+1/2} = \frac{1}{\pi_j}(F_j(\boldsymbol{\omega}_{t+1/2}) - \boldsymbol{\alpha}_j^{(t)}) + \bar{\boldsymbol{\alpha}}^{(t)}$ to get

$$\begin{aligned}\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2 &= \|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}_t) - \boldsymbol{\alpha}_i^{(t)}) - \tfrac{1}{n\pi_j}(F_j(\boldsymbol{\omega}_{t+1/2}) - \boldsymbol{\alpha}_j^{(t)})\|^2 &(41)\\ &= \|\tfrac{1}{n\pi_i}(F_i(\boldsymbol{\omega}_t) - \boldsymbol{\alpha}_i^{(t)}) + \tfrac{1}{n\pi_j}(F_j(\boldsymbol{\omega}_t) - F_j(\boldsymbol{\omega}_{t+1/2})) + \tfrac{1}{n\pi_j}(\boldsymbol{\alpha}_j^{(t)} - F_j(\boldsymbol{\omega}_t))\|^2 &(42)\\ &\leq \frac{3}{n^2\pi_i^2}\|F_i(\boldsymbol{\omega}_t) - \boldsymbol{\alpha}_i^{(t)}\|^2 + \frac{3}{n^2\pi_j^2}\|F_j(\boldsymbol{\omega}_t) - F_j(\boldsymbol{\omega}_{t+1/2})\|^2 + \frac{3}{n^2\pi_j^2}\|\boldsymbol{\alpha}_j^{(t)} - F_j(\boldsymbol{\omega}_t))\|^2\,. &(43)\end{aligned}$$

Then, we just need to notice that since i and j are both sampled from the same distribution,

$$\mathbb{E}[\tfrac{1}{n^2\pi_j^2}\|F_j(\boldsymbol{\omega}_t) - \boldsymbol{\alpha}_j^{(t)}\|^2] = \mathbb{E}[\tfrac{1}{n^2\pi_i^2}\|F_i(\boldsymbol{\omega}_t) - \boldsymbol{\alpha}_i^{(t)}\|^2]\,. \tag{44}$$

By assuming that each F_i is L_i-Lipschitz we get,

$$\begin{aligned}\mathbb{E}[\tfrac{1}{n^2\pi_j^2}\|F_j(\boldsymbol{\omega}_t) - F_j(\boldsymbol{\omega}_{t+1/2})\|^2] &= \frac{1}{n^2}\sum_{j=1}^n \pi_j\mathbb{E}[\|F_j(\boldsymbol{\omega}_t) - F_j(\boldsymbol{\omega}_{t+1/2})\|^2] &(45)\\ &\leq \frac{1}{n^2}\sum_{j=1}^n \frac{L_j^2}{\pi_j}\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2] &(46)\\ &= \bar{L}^2\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2\,, &(47)\end{aligned}$$

where $\bar{L}^2 := \frac{1}{n^2}\sum_{i=1}^n \frac{L_i^2}{\pi_j}$. Note that $\boldsymbol{\omega}_t$ and $\boldsymbol{\omega}_{t+1/2}$ do not depend on j (which is the index for the update step), that is not the case for i (the index for the extrapolation step) since $\boldsymbol{\omega}_{t+1/2}$ is the result of the extrapolation. □

Lemma 5. *Let $(\boldsymbol{\alpha}_j^{(t)})$ be updated according to the rules of a q-uniform memorization algorithm (6). Let us note $H_t := \frac{1}{n}\sum_{i=1}^n \frac{1}{n\pi_i}\|F_i(\boldsymbol{\omega}_t) - \boldsymbol{\alpha}_i^{(t)}\|^2$ For any $t \in \mathbb{N}$,*

$$\mathbb{E}[H_{t+1}] = 4\bar{L}^2\frac{q}{n}(\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2] + \eta_t^2\mathbb{E}[\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2]) + \frac{2n-q}{2n}\mathbb{E}[H_t]\,. \tag{48}$$

Proof. We will use the definition of q-uniform memorization algorithms (saying that $\boldsymbol{\alpha}_i$ is updated with probability q/n). We call this event "i updated",

$$\begin{aligned}
\mathbb{E}[H_{t+1}] &:= \mathbb{E}[\frac{1}{n}\sum_{i=1}^{n}\tfrac{1}{n\pi_i}\|\boldsymbol{\alpha}_i^{(t+1)} - F_i(\boldsymbol{\omega}_{t+1})\|^2] \\
&= \frac{1}{n}\mathbb{E}[\sum_{i \text{ updated}}\tfrac{1}{n\pi_i}\|\boldsymbol{\alpha}_i^{(t+1)} - F_i(\boldsymbol{\omega}_{t+1})\|^2 + \sum_{i \text{ not updated}}\tfrac{1}{n\pi_i}\|\boldsymbol{\alpha}_i^{(t+1)} - F_i(\boldsymbol{\omega}_{t+1})\|^2] \\
&= \frac{1}{n}\mathbb{E}[\sum_{i \text{ updated}}\tfrac{1}{n\pi_i}\|F_i(\boldsymbol{\omega}_t) - F_i(\boldsymbol{\omega}_{t+1})\|^2 + \sum_{i \text{ not updated}}\tfrac{1}{n\pi_i}\|\boldsymbol{\alpha}_i^{(t)} - F_i(\boldsymbol{\omega}_{t+1})\|^2] \\
&\leq \frac{1}{n}\mathbb{E}[\sum_{i \text{ updated}}\tfrac{1}{n\pi_i}L_i^2\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1}\|^2] + \frac{1}{n}\mathbb{E}[\sum_{i \text{ not updated}}\tfrac{1}{n\pi_i}\|\boldsymbol{\alpha}_i^{(t)} - F_i(\boldsymbol{\omega}_{t+1})\|^2] \\
&= \frac{1}{n}\sum_i \tfrac{1}{n\pi_i}L_i^2\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1}\|^2]\mathbb{P}(i \text{ updated}) + \frac{1}{n}\sum_i \tfrac{1}{n\pi_i}\mathbb{P}(i \text{ not updated})\mathbb{E}[\|\boldsymbol{\alpha}_i^{(t)} - F_i(\boldsymbol{\omega}_{t+1})\|^2] \\
&= \frac{q}{n}\bar{L}^2\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1}\|^2] + \frac{n-q}{n}\mathbb{E}[\tfrac{1}{n}\sum_{i=1}^{n}\tfrac{1}{n\pi_i}\|\boldsymbol{\alpha}_i^{(t)} - F_i(\boldsymbol{\omega}_{t+1})\|^2] \\
&\leq \frac{q}{n}\bar{L}^2(2\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2] + 2\mathbb{E}[\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_{t+1}\|^2]) + \frac{n-q}{n}\mathbb{E}[(1+\beta)H_t + \frac{1+\beta}{n\beta}\sum_{i=1}^{n}\tfrac{1}{n\pi_i}\|F_i(\boldsymbol{\omega}_t) - F_i(\boldsymbol{\omega}_{t+1})\|^2]\,,
\end{aligned}$$

where $\beta > 0$ is not fixed yet comes from $\|\boldsymbol{a}+\boldsymbol{b}\|^2 \leq (1+\beta)\|\boldsymbol{a}\|^2 + (1+1/\beta)\|\boldsymbol{b}\|^2$. Now, let us recall that we defined $\bar{L}^2 := \frac{1}{n}\sum_{i=1}^{n}\frac{1}{n\pi_i}L_i^2$ to get,

$$\begin{aligned}
\mathbb{E}[H_{t+1}] &\leq \frac{q}{n}\bar{L}^2(2\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2] + 2\eta_t^2\mathbb{E}[\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2]) + \frac{n-q}{n}\left((1+\beta)\mathbb{E}[H_t] + \frac{\beta}{1+\beta}\mathbb{E}[\bar{L}^2\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1}\|^2]\right) \\
&\leq \frac{q}{n}\bar{L}^2\big(2\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2] + 2\eta_t^2\mathbb{E}[\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2]\big) \\
&\quad + \frac{n-q}{n}\left((1+\beta)\mathbb{E}[H_t] + 2\frac{\beta}{1+\beta}\mathbb{E}[\bar{L}^2\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2 + 2\frac{\beta}{1+\beta}\mathbb{E}[\bar{L}^2\eta_t^2\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2]\right) \\
&= 2\bar{L}^2\left(\frac{q}{n} + \beta\frac{n-q}{(1+\beta)n}\right)\big(\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2] + \eta_t^2\mathbb{E}[\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2]\big) + (1+\beta)\frac{n-q}{n}\mathbb{E}[H_t]\,.
\end{aligned}$$

Now we will chose $\beta = \frac{q}{2n-q} > 0$ to get,

$$\mathbb{E}[H_{t+1}] \leq 4\frac{q}{n}\bar{L}^2\left(\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2] + \eta_t^2\mathbb{E}[\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2]\right) + \frac{2n-q}{2n}\mathbb{E}[H_t]\,. \tag{49}$$

□

Theorem' 3. *Under Assumption 1, after t iterations, the iterate $\boldsymbol{\omega}_t$ computed by a q-memorization algorithm with step-sizes $(\eta_\theta, \eta_\phi) = \left((12\bar{L}_{\boldsymbol{\theta}})^{-1}, (12\bar{L}_{\boldsymbol{\varphi}})^{-1}\right)$ verifies:*

$$\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] \leq \left(1 - \min\left\{\frac{\mu_{\boldsymbol{\theta}}}{12\bar{L}_{\boldsymbol{\theta}}}, \frac{\mu_{\boldsymbol{\varphi}}}{12\bar{L}_{\boldsymbol{\varphi}}}, \frac{q}{12n}\right\}\right)^t \mathbb{E}[\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}^*\|_2^2]\,. \tag{50}$$

Proof. In this proof we will consider a constant step-size $\eta_t = (\eta_{\boldsymbol{\theta}}, \eta_\phi)$. For simplicity of notations we will consider the notation,

$$\begin{gathered}
\bar{L}^2\|\boldsymbol{\omega}\|^2 := \bar{L}_{\boldsymbol{\theta}}^2\|\boldsymbol{\theta}\|^2 + \bar{L}_{\boldsymbol{\varphi}}^2\|\boldsymbol{\varphi}\|^2\,, \quad \eta^2\|\boldsymbol{\omega}\|^2 := \eta_{\boldsymbol{\theta}}^2\|\boldsymbol{\theta}\|^2 + \eta_{\boldsymbol{\varphi}}^2\|\boldsymbol{\varphi}\|^2\,, \quad \mu\|\boldsymbol{\omega}\|^2 := \mu_{\boldsymbol{\theta}}^2\|\boldsymbol{\theta}\|^2 + \mu_{\boldsymbol{\varphi}}^2\|\boldsymbol{\varphi}\|^2 \\
\eta\mu = (\eta_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}, \eta_{\boldsymbol{\varphi}}\mu_{\boldsymbol{\varphi}})\,, \quad \sigma\bar{L}^2 = (\sigma_{\boldsymbol{\theta}}\bar{L}_{\boldsymbol{\theta}}^2, \sigma_{\boldsymbol{\varphi}}\bar{L}_{\boldsymbol{\varphi}}^2) \quad \text{and} \quad \eta^2\bar{L}^2 = (\eta_{\boldsymbol{\theta}}^2\bar{L}_{\boldsymbol{\theta}}^2, \eta_{\boldsymbol{\varphi}}^2\bar{L}_{\boldsymbol{\varphi}}^2)\,.
\end{gathered}$$

Let us first consider the case where the operator is strongly monotone, in that case one can combine Lemma 3 for $\boldsymbol{\omega} = \boldsymbol{\omega}^*$ with Lemma 2 for $\boldsymbol{\omega} = \boldsymbol{\omega}_t$ and $\boldsymbol{\omega}' = \boldsymbol{\omega}_{t+1/2}$ to get,

$$\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}^*\|_2^2 \leq (1-\eta\mu)\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2 - (1-2\eta\mu)\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 + \eta^2\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|_2^2\,. \tag{51}$$

Let us define $\mathcal{L}_t := \mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] + \sigma\mathbb{E}[H_t]$, where $H_t := \frac{1}{n}\sum_{i=1}^n \|F_i(\boldsymbol{\omega}_t) - \boldsymbol{\alpha}_i^{(t)}\|^2$. We can combine (51) with Lemma 5 multiplied by a constant $\sigma > 0$ that we will set later to get

$$\begin{aligned}
\mathcal{L}_{t+1} &= \mathbb{E}[\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}^*\|_2^2] + \sigma\mathbb{E}[H_{t+1}] && (52)\\
&\leq (1-\eta\mu)\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2 - (1-2\eta\mu)\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2 + \eta^2\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|_2^2 \\
&\quad + \sigma\left(\tfrac{4q}{n}\bar{L}^2(\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1/2}\|^2] + \eta^2\mathbb{E}[\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2]) + \tfrac{2n-q}{2n}\mathbb{E}[H_t]\right) && (53)\\
&= (1-\eta\mu)\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - (1-2\eta\mu - \sigma\tfrac{4q}{n}\bar{L}^2)\mathbb{E}[\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2] \\
&\quad + \eta^2(1+\sigma\tfrac{4q}{n}\bar{L}^2)\mathbb{E}[\|\boldsymbol{g}_t - \boldsymbol{g}_{t+1/2}\|^2]) + \tfrac{2n-q}{2n}\sigma\mathbb{E}[H_t]\,. && (54)
\end{aligned}$$

We can then use Lemma 4 to get

$$\begin{aligned}
\mathcal{L}_{t+1} &\leq (1-\eta\mu)\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - (1-2\eta\mu - \tfrac{4q}{n}\sigma\bar{L}^2 - 3\eta^2\bar{L}^2(1+\tfrac{4q}{n}\sigma\bar{L}^2))\mathbb{E}[\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2] \\
&\quad + \left(6\tfrac{\eta^2}{\sigma}(1+\tfrac{4q}{n}\sigma\bar{L}^2) + \tfrac{2n-q}{2n}\right)\sigma\mathbb{E}[H_t]\,. && (55)
\end{aligned}$$

Then, setting $\sigma = \frac{24n\eta^2}{q}$ we get

$$\begin{aligned}
\mathcal{L}_{t+1} &\leq (1-\eta\mu)\mathbb{E}[\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2] - (1-2\eta\mu - 96\eta^2\bar{L}^2 - 3\eta^2\bar{L}^2(1+96\eta^2\bar{L}^2))\mathbb{E}[\|\boldsymbol{\omega}_{t+1/2} - \boldsymbol{\omega}_t\|_2^2] \\
&\quad + \left(\tfrac{q}{4n}(1+96\eta^2\bar{L}^2) + \tfrac{2n-q}{2n}\right)\sigma\mathbb{E}[H_t]\,. && (56)
\end{aligned}$$

With $\eta \leq \frac{1}{12\bar{L}}$ we obtain that,

$$1 - 2\eta\mu - 96\eta^2\bar{L}^2 - 3\eta^2\bar{L}^2(1+96\eta^2\bar{L}^2) \geq 1 - \tfrac{2\mu}{12\bar{L}} - \tfrac{96}{144} - \tfrac{3}{144}(1+\tfrac{96}{144}) \geq 1 - \tfrac{2}{12} - \tfrac{96}{144} - \tfrac{3}{144}(1+\tfrac{96}{144}) > 0\,, \quad (57)$$

where we used that $\frac{\mu}{\bar{L}} \leq 1$. Thus for $\eta = \frac{1}{12\bar{L}}$ we have

$$\mathcal{L}_{t+1} \leq \left(1 - \min\left\{\frac{\mu}{12\bar{L}}, \frac{q}{12n}\right\}\right)\mathcal{L}_t = \left(1 - \frac{1}{12}\min\left\{\frac{\mu_{\boldsymbol{\theta}}}{\bar{L}_{\boldsymbol{\theta}}}, \frac{\mu_{\boldsymbol{\varphi}}}{\bar{L}_{\boldsymbol{\varphi}}}, \frac{q}{n}\right\}\right)\mathcal{L}_t\,. \quad (58)$$

Recall that $\mu := (\mu_{\boldsymbol{\theta}}, \mu_{\boldsymbol{\varphi}})$ and $\bar{L} := (\bar{L}_{\boldsymbol{\theta}}, \bar{L}_{\boldsymbol{\varphi}})$.

□

C. Details on the SVRE–GAN Algorithm

C.1. Practical Aspect

Particular choices such as the optimization method (*e.g.* Adam (Kingma and Ba, 2015)), learning rates, and normalization, have been established in practice as almost *prerequisite* for convergence[4], in contrast to supervised classification problems where they have been shown to only provide a marginal value (Wilson et al., 2017). To our knowledge, SVRE is the only method that works with constant step size for GANs on non–trivial datasets. This combined with the fact that recent works empirically tune the first moment controlling hyperparameter to 0 (β_1, see below) and the variance reduction (VR) one (β_2, see below) to a non-zero value, sheds light on the reason behind the success of Adam on GANs.

However, combining SVRE with adaptive step size scheme on GANs remains an open problem. We first briefly describe the update rule of Adam, and then we propose a new adaptation of it that is more suitable for VR methods, which we refer to as variance reduced Adam (VRAd).

Adam. Adam stores an exponentially decaying average of both past gradients m_t and squared gradients v_t, for each parameter of the model:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t \quad (59)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2\,, \quad (60)$$

[4]For instance, Daskalakis et al. (2018); Gidel et al. (2019) plugged Adam into their principled method to get better results.

Algorithm 2 Pseudocode for SVRE-GAN.

Input: dataset $\mathcal{D}$, noise dataset $\mathcal{Z}$ ($|\mathcal{Z}| = |\mathcal{D}| = n$), stopping iteration T, learning rates η_D, η_G, generator loss $\mathcal{L}^G$, discriminator loss $\mathcal{L}^D$, mini-batch size B.
Initialize: D, G
for $e = 0$ **to** $T-1$ **do**
  $D^{\mathcal{S}} = D$ and $\boldsymbol{\mu}_D = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\nabla_D \mathcal{L}^{(D)}(G^{\mathcal{S}}, D^{\mathcal{S}}, \mathcal{D}_j, \mathcal{Z}_i)$
  $G^{\mathcal{S}} = G$ and $\boldsymbol{\mu}_G = \frac{1}{n}\sum_{i=1}^{n}\nabla_G \mathcal{L}_i^{(G)}(G^{\mathcal{S}}, D^{\mathcal{S}})$
  $N \sim \text{Geom}\left(B/n\right)$ (length of the epoch)
  for $i = 0$ **to** $N-1$ **do**
    Sample mini-batches (n_d, n_z); do **extrapolation:**
    $\tilde{D} = D - \eta_D \boldsymbol{d}_D(G, D, G^{\mathcal{S}}, D^{\mathcal{S}}, n_z)$ ▷ (66)
    $\tilde{G} = G - \eta_G \boldsymbol{d}_G(G, D, G^{\mathcal{S}}, D^{\mathcal{S}}, n_d, n_z)$ ▷ (67)
    Sample new mini-batches (n_d, n_z); do **update:**
    $D = D - \eta_D \boldsymbol{d}_D(\tilde{G}, \tilde{D}, G^{\mathcal{S}}, D^{\mathcal{S}}, n_z)$ ▷ (66)
    $G = G - \eta_G \boldsymbol{d}_G(\tilde{G}, \tilde{D}, G^{\mathcal{S}}, D^{\mathcal{S}}, n_d, n_z)$ ▷ (67)
  end for
end for
Output: G, D

where $\beta_1, \beta_2 \in [0, 1]$, $m_0 = 0$, $v_0 = 0$, and $t = 1, \dots T$ denotes the iteration. m_t and v_t are respectively the estimates of the first and the second moments of the stochastic gradient. To compensate the bias toward 0 due to initialization, Kingma and Ba (2015) propose to use bias-corrected estimates of these first two moments:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{61}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{62}$$

The Adam update rule can be described as:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon}. \tag{63}$$

Adam can be understood as an approximate gradient method with a diagonal step size of $\eta_{Adam} := \frac{\eta}{\sqrt{\boldsymbol{v}_t} + \epsilon}$. Since VR methods aim to provide a vanishing $\boldsymbol{v}_t$, they lead to a too large step-size η_{Adam} of $\frac{\eta}{\epsilon}$. This could indicate that the update rule of Adam may not be a well-suited method to combine with VR methods.

VRAd. This motivates the introduction of a new Adam-inspired variant of adaptive step sizes that maintain a reasonable size even when $\boldsymbol{v}_t$ vanishes,

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta \frac{|\hat{\boldsymbol{m}}_t|}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon} \hat{\boldsymbol{m}}_t \,. \tag{VRAd}$$

This adaptive variant of Adam is motivated by the step size $\eta^* = \eta \frac{\boldsymbol{m}_t^2}{\boldsymbol{v}_t}$ derived by Schaul et al. (2013). (VRAd) is simply the square-root of η^* in order to stick with Adam's scaling of $\boldsymbol{v}_t$.

C.2. SVRE-GAN

In order to cope with the issues introduced by the stochastic game formulation of the GAN models, we proposed the SVRE algorithm Alg. 1 which combines SVRG and extragradient method. We refer to the method of applying SVRE to train GANs as the *SVRE-GAN* method, and we describe it in details in Alg. 2 (generalizing it with mini-batching, but using uniform probabilities). Assuming that we have $\mathcal{D}[n_d]$ and $\mathcal{Z}[n_z]$, respectively two mini-batches of size B of the true dataset and the noise dataset, we compute $\nabla_D \mathcal{L}^D(G, D, \mathcal{D}[n_d], \mathcal{Z}[n_z])$ and $\nabla_G \mathcal{L}^G(G, D, \mathcal{Z}[n_z])$ the respective mini-batches gradient of

Algorithm 3 Pseudocode for Restarted SVRE.

1: **Input:** Stopping time T, learning rates $\eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{\varphi}}$, both players' losses $\mathcal{L}^{(\boldsymbol{\theta})}$ and $\mathcal{L}^{(\boldsymbol{\varphi})}$, probability of restart p.
2: **Initialize:** $\boldsymbol{\varphi}, \boldsymbol{\theta}, t = 0$ ▷ t is for the online average computation.
3: **for** $e = 0$ **to** $T-1$ **do**
4: Draw $\texttt{restart} \sim \mathrm{B}(p)$. ▷ Check if we restart the algorithm.
5: **if** $\texttt{restart}$ **and** $e > 0$ **then**
6: $\boldsymbol{\varphi} \leftarrow \bar{\boldsymbol{\varphi}}$, $\boldsymbol{\theta} \leftarrow \bar{\boldsymbol{\theta}}$ and $t = 1$
7: **end if**
8: $\boldsymbol{\varphi}^{\mathcal{S}} \leftarrow \boldsymbol{\varphi}$ and $\boldsymbol{\mu}_{\boldsymbol{\varphi}}^{\mathcal{S}} \leftarrow \frac{1}{|\mathcal{Z}|} \sum_{i=1}^{n} \nabla_{\boldsymbol{\varphi}} \mathcal{L}_i^{(\boldsymbol{\varphi})}(\boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$
9: $\boldsymbol{\theta}^{\mathcal{S}} \leftarrow \boldsymbol{\theta}$ and $\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\mathcal{S}} \leftarrow \frac{1}{|\boldsymbol{\varphi}|} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i^{(\boldsymbol{\theta})}(\boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$
10: $N \sim \mathrm{Geom}\left(1/n\right)$ ▷ Length of the epoch.
11: **for** $i = 0$ **to** $N-1$ **do**
12: **Sample** $i_{\boldsymbol{\theta}} \sim \pi_{\boldsymbol{\theta}}$, $i_{\boldsymbol{\varphi}} \sim \pi_{\boldsymbol{\varphi}}$, do **extrapolation:**
13: $\tilde{\boldsymbol{\varphi}} \leftarrow \boldsymbol{\varphi} - \eta_{\boldsymbol{\theta}} \boldsymbol{d}_{\boldsymbol{\varphi}}(\boldsymbol{\theta}, \boldsymbol{\varphi}, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$, $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} - \eta_{\boldsymbol{\varphi}} \boldsymbol{d}_{\boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\varphi}, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$ ▷ (6) and (7)
14: **Sample** $i_{\boldsymbol{\theta}} \sim \pi_{\boldsymbol{\theta}}$, $i_{\boldsymbol{\varphi}} \sim \pi_{\boldsymbol{\varphi}}$, do **update:**
15: $\boldsymbol{\varphi} \leftarrow \boldsymbol{\varphi} - \eta_{\boldsymbol{\theta}} \boldsymbol{d}_{\boldsymbol{\varphi}}(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\varphi}}, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$, $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_{\boldsymbol{\varphi}} \boldsymbol{d}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\varphi}}, \boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}})$ ▷ (6) and (7)
16: $\bar{\boldsymbol{\theta}} \leftarrow \frac{t}{t+1} \bar{\boldsymbol{\theta}} + \frac{1}{t+1} \boldsymbol{\theta}$ and $\bar{\boldsymbol{\varphi}} \leftarrow \frac{t}{t+1} \bar{\boldsymbol{\varphi}} + \frac{1}{t+1} \boldsymbol{\varphi}$ ▷ Online computation of the average.
17: $t \leftarrow t + 1$ ▷ Increment t for the online average computation.
18: **end for**
19: **end for**
20: **Output:** $\boldsymbol{\theta}, \boldsymbol{\varphi}$

the discriminator and the generator:

$$\nabla_D \mathcal{L}^D(G, D, \mathcal{D}[n_d], \mathcal{Z}[n_z]) := \frac{1}{|n_z|} \frac{1}{|n_d|} \sum_{i \in n_z} \sum_{j \in n_d} \nabla_D \mathcal{L}^D(G, D, \mathcal{D}_j, \mathcal{Z}_i) \tag{64}$$

$$\nabla_G \mathcal{L}^G(G, D, \mathcal{Z}[n_z]) := \frac{1}{|n_z|} \sum_{i \in n_z} \nabla_G \mathcal{L}^G(G, D, \mathcal{Z}_i)\,, \tag{65}$$

where $\mathcal{Z}_i$ and $\mathcal{D}_j$ are respectively the i^{th} example of the noise dataset and the j^{th} of the true dataset. Note that n_z and n_d are lists and thus that we allow repetitions in the summations over n_z and n_d. The variance reduced gradient of the SVRG method are thus given by:

$$\boldsymbol{d}_D(G, D, G^{\mathcal{S}}, D^{\mathcal{S}}) := \boldsymbol{\mu}_D + \nabla_D \mathcal{L}^D(G, D, \mathcal{D}[n_d], \mathcal{Z}[n_z]) - \nabla_D \mathcal{L}^D(G^{\mathcal{S}}, D^{\mathcal{S}}, \mathcal{D}[n_d], \mathcal{Z}[n_z]) \tag{66}$$

$$\boldsymbol{d}_G(G, D, G^{\mathcal{S}}, D^{\mathcal{S}}) := \boldsymbol{\mu}_G + \nabla_G \mathcal{L}^G(G, D, \mathcal{Z}[n_z]) - \nabla \mathcal{L}^G(G^{\mathcal{S}}, D^{\mathcal{S}}, \mathcal{Z}[n_z])\,, \tag{67}$$

where $G^{\mathcal{S}}$ and $D^{\mathcal{S}}$ are the snapshots and $\boldsymbol{\mu}_D$ and $\boldsymbol{\mu}_G$ their respective gradients.

Alg. 2 summarizes the SVRG optimization extended to GAN. To obtain that $\mathbb{E}\left[\nabla_{\boldsymbol{\Theta}^{\mathcal{S}}} \mathcal{L}(\boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\varphi}^{\mathcal{S}}, \cdot) - \boldsymbol{\mu}\right]$ vanishes, when updating $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ where the expectation is over samples of $\mathcal{D}$ and $\mathcal{Z}$ respectively, we use the snapshot networks $\boldsymbol{\theta}^{\mathcal{S}}$ and $\boldsymbol{\varphi}^{\mathcal{S}}$ for the second term in lines $9, 10, 12$ & 13. Moreover, the noise dataset $\mathcal{Z} \sim p_z$, where $|\mathcal{Z}| = |\mathcal{D}| = n$, is fixed. Empirically we observe that directly sampling from p_z (contrary to fixing the noise dataset and re-sampling it with frequency m) does not impact the performance, as $|\mathcal{Z}|$ is usually high.

Note that the double sum in Line 4 can actually be written as two simple sums because of the separability of the expectations in the standard GAN objectives. Thus the time complexity for calculating μ^D is still $O(n)$ and not $O(n^2)$ which would be prohibitively expensive.

D. Restarted SVRE

Alg. 3 describes the restarted version of SVRE presented in § 4.3. With a probability p (fixed) before the computation of $\boldsymbol{\mu}_{\boldsymbol{\varphi}}^{\mathcal{S}}$ and $\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\mathcal{S}}$, we decide whether to restart SVRE (by using the averaged iterate as the new starting point (Alg. 3, Line 6) $\bar{\boldsymbol{\omega}}_t$) or computing the batch snapshot at a point $\boldsymbol{\omega}_t$.

E. Details on the implementation

For our experiments, we used the PyTorch[5] deep learning framework, whereas for computing the FID and IS metrics, we used the provided implementations in Tensorflow[6].

E.1. Metrics

We provide more details about the metrics enumerated in § 5. Both FID and IS use: (i) the *Inception v3 network* (Szegedy et al., 2015) that has been trained on the ImageNet dataset consisting of ∼1 million RGB images of 1000 classes, $C = 1000$. (ii) a sample of m generated images $x \sim p_g$, where usually $m = 50000$.

Given an image x, IS uses the softmax output of the Inception network $p(y|x)$ which represents the probability that x is of class $c_i, i \in 1 \dots C$, i.e., $p(y|x) \in [0,1]^C$. It then computes the marginal class distribution $p(y) = \int_x p(y|x)p_g(x)$. IS measures the Kullback–Leibler divergence $\mathbb{D}_{KL}$ between the predicted conditional label distribution $p(y|x)$ and the marginal class distribution $p(y)$. More precisely, it is computed as follows:

$$IS(G) = \exp\left(\mathbb{E}_{x\sim p_g}[\mathbb{D}_{KL}(p(y|x)||p(y))]\right) = \exp\left(\frac{1}{m}\sum_{i=1}^{m}\sum_{c=1}^{C} p(y_c|x_i)\log\frac{p(y_c|x_i)}{p(y_c)}\right). \tag{68}$$

It aims at estimating (i) if the samples look realistic i.e., $p(y|x)$ should have low entropy, and (ii) if the samples are diverse (from different ImageNet classes) i.e., $p(y)$ should have high entropy. As these are combined using the Kullback–Leibler divergence, the higher the score is, the better the performance. Note that the range of IS scores at convergence varies across datasets, as the Inception network is pretrained on the ImageNet classes. For example, we obtain low IS values on the SVHN dataset as a large fraction of classes are numbers, which typically do not appear in the ImageNet dataset. Since **MNIST** has greyscale images, we used a classifier trained on this dataset and used $m = 5000$. For the rest of the datasets, we used the original implementation[7] of IS in TensorFlow, and $m = 50000$.

Contrary to IS, FID aims at comparing the synthetic samples $x \sim p_g$ with those of the training dataset $x \sim p_d$ in a feature space. The samples are embedded using the first several layers of the Inception network. Assuming p_g and p_d are multivariate normal distributions, it then estimates the means $\boldsymbol{m}_g$ and $\boldsymbol{m}_d$ and covariances C_g and C_d, respectively for p_g and p_d in that feature space. Finally, FID is computed as:

$$\mathbb{D}_{\text{FID}}(p_d, p_g) \approx d^2((\boldsymbol{m}_d, C_d), (\boldsymbol{m}_g, C_g)) = ||\boldsymbol{m}_d - \boldsymbol{m}_g||_2^2 + Tr(C_d + C_g - 2(C_d C_g)^{\frac{1}{2}}), \tag{69}$$

where d^2 denotes the Fréchet Distance. Note that as this metric is a distance, the lower it is, the better the performance. We used the original implementation of FID[8] in Tensorflow, along with the provided statistics of the datasets.

To evaluate SVRE effectively, we used the **second moment estimate** (uncentered variance, see § C.1) of the gradient estimate throughout the iterations $t = 1 \dots T$ per parameter, computed as: $v_t = \gamma v_{t-1} + (1-\gamma)g_t^2$, where g_t denotes the gradient estimate for the parameter and iteration t, and $\gamma = 0.9$. For SVRE, g_t is $d_{\boldsymbol{\varphi}}$ and $d_{\boldsymbol{\theta}}$ (see Eq. 6 & 7) for G and D, respectively. We initialize $g_0 = 0$ and we use bias-corrected estimates: $\hat{v} = \frac{v_t}{1-\gamma^t}$.

For the experiments on **MNIST** in § 5, we plot in § F the **entropy** (E) of the generated samples' class distribution, as well as the **total variation** (TV) between the class distribution of the generated samples and a uniform one (both computed using a pretrained network that classifies its 10 classes).

E.2. Architectures & Hyperparameters

MNIST. For experiments on the **MNIST** dataset, we used the DCGAN architectures, listed in Table 3, and the parameters are initialized using PyTorch default initialization. We used mini-batch sizes of 50 samples, whereas for full dataset passes we used mini-batches of 500 samples as this reduces the wall-clock time for its computation. For experiments on this dataset, we used the original GAN loss (Goodfellow et al., 2014). For both the baseline and the SVRE variants we tried the following step sizes $\eta = [1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}]$. We observed that SVRE can be used with larger step sizes. In Table 2, we used $\eta = 1 \times 10^{-4}$ and $\eta = 1 \times 10^{-2}$ for SE–A and SVRE(–VRAd), respectively.

[5] https://pytorch.org/
[6] https://www.tensorflow.org/
[7] https://github.com/openai/improved-gan/
[8] https://github.com/bioinf-jku/TTUR

Generator
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
transposed conv. (kernel: 3×3, $128 \to 512$; stride: 1)
Batch Normalization
ReLU
transposed conv. (kernel: 4×4, $512 \to 256$, stride: 2)
Batch Normalization
ReLU
transposed conv. (kernel: 4×4, $256 \to 128$, stride: 2)
Batch Normalization
ReLU
transposed conv. (kernel: 4×4, $128 \to 1$, stride: 2, pad: 1)
$Tanh(\cdot)$

Discriminator
Input: $x \in \mathbb{R}^{1\times28\times28}$
conv. (kernel: 4×4, $1 \to 64$; stride: 2; pad:1)
LeakyReLU (negative slope: 0.2)
conv. (kernel: 4×4, $64 \to 128$; stride: 2; pad:1)
Batch Normalization
LeakyReLU (negative slope: 0.2)
conv. (kernel: 4×4, $128 \to 256$; stride: 2; pad:1)
Batch Normalization
LeakyReLU (negative slope: 0.2)
conv. (kernel: 3×3, $256 \to 1$; stride: 1)
$Sigmoid(\cdot)$

Table 3: Architectures used for experiments on **MNIST**.

Self–Attention Block (d **– input depth)**		
Input: $t \in \mathbb{R}^{d\times H\times W}$		
i: conv. (kernel: 1×1, $d \to \lfloor d/8 \rfloor$)	*ii:* conv. (kernel: 1×1, $d \to \lfloor d/8 \rfloor$)	*iii:* conv. (kernel: 1×1, $d \to d$)
iv: softmax(*(i)* $\otimes$ *(ii)*)		
Output: $\gamma\big(\textit{(iv)} \otimes \textit{(iii)}\big) + t$		

Table 4: Layers of the self–attention block used in the SAGAN architectures (see Tables 5 & 6), where $\otimes$ denotes matrix multiplication and γ is a scale parameter initialized with 0. The columns emphasize that the execution is in parallel, more preciesly, that the block input t is input to the convolutional layers *(i)–(iii)*. The shown row ordering corresponds to consecutive layers' order, *e.g.* softmax is done on the product of the outputs of the *(i)* & *(ii)* convolutional layers. The 1×1 convolutional layers have stride of 1. For complete details see (Zhang et al., 2018).

Shallower SAGAN architectures. Most of our experiments were done using the SAGAN architectures (Zhang et al., 2018), as the techniques of self-attention introduced in SAGAN were used to obtain the state-of-art GAN results on ImageNet (Brock et al., 2019). In summary, these architectures: (i) allow for attention-driven, long-range dependency modeling, (ii) use spectral normalization (Miyato et al., 2018) on both G and D (efficiently computed with the *power iteration* method); and (iii) use different learning rates for G and D, as advocated in (Heusel et al., 2017). The foremost is obtained by combining weights, or alternatively attention vectors with the convolutions across layers, so as to allow modeling textures that are consistent globally–for the generator, or enforcing geometric constraints on the global image structure–for the discriminator. We found deeper architectures less stable, as the training often fails to converge. As of this reason and lack of computational resources, our experiments were designed to compare SVRE with its stochastic counterpart on a fixed experimental setup, contrary to obtaining state of the art performances. We used shallower SAGAN architectures that reduce computation and memory footprint of the model. In particular, we used the architectures listed in Table 5 for **CIFAR-10** and **SVHN** datasets, and the architectures described in Table 6 for experiments on **ImageNet**. The models' parameters are initialized using PyTorch default initialization.

For experiments with SAGAN, we used the hinge version of the adversarial non-saturating loss (Lim and Ye, 2017; Zhang et al., 2018):

$$\mathcal{L}_D = \mathbb{E}_{x\sim p_d} \max(0, 1 - D(x)) + \mathbb{E}_{z\sim p_z} \max(0, 1 + D(G(z)) \tag{70}$$

$$\mathcal{L}_G = -\mathbb{E}_{z\sim p_z} D(G(z)). \tag{71}$$

For the SE–A baseline we obtained best performances when $\eta_G = 1 \times 10^{-4}$ and $\eta_D = 4 \times 10^{-4}$, for G and D, respectively. Similarly as noted for **MNIST**, using SVRE allows for using larger order of the step size on the rest of the datasets, whereas

Generator
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
transposed conv. (kernel: 4×4, $128 \to 256$; stride: 1)
Spectral Normalization
Batch Normalization
ReLU
transposed conv. (kernel: 4×4, $256 \to 128$, stride: 2, pad: 1)
Spectral Normalization
Batch Normalization
ReLU
Self–Attention Block (128)
transposed conv. (kernel: 4×4, $128 \to 64$, stride: 2, pad: 1)
Batch Normalization
ReLU
Self–Attention Block (64)
transposed conv. (kernel: 4×4, $64 \to 3$, stride: 2, pad: 1)
$Tanh(\cdot)$

Discriminator
Input: $x \in \mathbb{R}^{3\times32\times32}$
conv. (kernel: 4×4, $3 \to 64$; stride: 2; pad: 1)
Spectral Normalization
LeakyReLU (negative slope: 0.1)
conv. (kernel: 4×4, $64 \to 128$; stride: 2; pad: 1)
Spectral Normalization
LeakyReLU (negative slope: 0.1)
conv. (kernel: 4×4, $128 \to 256$; stride: 2; pad: 1)
Spectral Normalization
LeakyReLU (negative slope: 0.1)
Self–Attention Block (256)
conv. (kernel: 4×4, $256 \to 1$; stride: 1)

Table 5: Shallower SAGAN architectures for experiments on **SVHN** and **CIFAR-10**. The self-attention block is described in Table 4. We use the default PyTorch hyperparameters for the Batch Normalization layer.

Generator
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$
transposed conv. (kernel: 4×4, $128 \to 512$; stride: 1)
Spectral Normalization
Batch Normalization
ReLU
transposed conv. (kernel: 4×4, $512 \to 256$, stride: 2, pad: 1)
Spectral Normalization
Batch Normalization
ReLU
transposed conv. (kernel: 4×4, $256 \to 128$, stride: 2, pad: 1)
Spectral Normalization
Batch Normalization
ReLU
Self–Attention Block (128)
transposed conv. (kernel: 4×4, $128 \to 64$, stride: 2, pad: 1)
Spectral Normalization
Batch Normalization
ReLU
Self–Attention Block (64)
transposed conv. (kernel: 4×4, $64 \to 3$, stride: 2, pad: 1)
$Tanh(\cdot)$

Discriminator
Input: $x \in \mathbb{R}^{3\times64\times64}$
conv. (kernel: 4×4, $3 \to 64$; stride: 2; pad: 1)
Spectral Normalization
LeakyReLU (negative slope: 0.1)
conv. (kernel: 4×4, $64 \to 128$; stride: 2; pad: 1)
Spectral Normalization
LeakyReLU (negative slope: 0.1)
conv. (kernel: 4×4, $128 \to 256$; stride: 2; pad: 1)
Spectral Normalization
LeakyReLU (negative slope: 0.1)
Self–Attention Block (256)
conv. (kernel: 4×4, $256 \to 512$; stride: 2; pad: 1)
Spectral Normalization
LeakyReLU (negative slope: 0.1)
Self–Attention Block (512)
conv. (kernel: 4×4, $512 \to 1$; stride: 1)

Table 6: Shallower SAGAN architectures for experiments on **ImageNet**. The self–attention block is described in Table 4. Relative to the architectures used for **SVHN** and **CIFAR-10** (see Table 5), the generator has one additional typical block (conv.–norm.–ReLU), whereas the discriminator both typical block and self–attention block (of more parameters).

G–ResBlock
Bypass:
Upsample($\times 2$)
Batch Normalization
ReLU
Upsample($\times 2$)
conv. (kernel: 3×3, $256 \to 256$; stride: 1; pad: 1)
Batch Normalization
ReLU
conv. (kernel: 3×3, $256 \to 256$; stride: 1; pad: 1)

D–ResBlock (ℓ–th block)
Bypass:
[AvgPool (kernel:2×2)], if $\ell = 1$
conv. (kernel: 1×1, $3_{\ell=1}/128_{\ell\neq1} \to 128$; stride: 1)
Spectral Normalization
[AvgPool (kernel:2×2, stride:2)], if $\ell \neq 1$
[ReLU], if $\ell \neq 1$
conv. (kernel: 3×3, $3_{\ell=1}/128_{\ell\neq1} \to 128$; stride: 1; pad: 1)
Spectral Normalization
ReLU
conv. (kernel: 3×3, $128 \to 128$; stride: 1; pad: 1)
Spectral Normalization
AvgPool (kernel:2×2)

Table 7: ResNet blocks used for the ResNet architectures (see Table 8). Each ResNet block contains skip connection (bypass), and a sequence of convolutional layers, normalization, and the ReLU non–linearity. The skip connection of the ResNet blocks for the Generator (left) upsamples the input using a factor of 2 (we use the default PyTorch upsampling algorithm–nearest neighbor), whose output is then added to the one obtained from the ResNet block listed above. The ResNet block for the Discriminator (right) differs if it is the first block in the network (following the input to the Discriminator), or a subsequent one, so as to avoid performing the ReLU non–linearity immediate on the input.

SE–A with increased step size ($\eta_G = 1 \times 10^{-3}$ and $\eta_D = 4 \times 10^{-3}$ failed to converge. In Table 2, $\eta_G = 1 \times 10^{-3}$, $\eta_D = 4 \times 10^{-3}$, and $\eta_G = 5 \times 10^{-3}$, $\eta_D = 8 \times 10^{-3}$, $\beta_1 = 0.3$ for SVRE and SVRE–VRAd, respectively. We did not use momentum for the vanilla SVRE experiments.

Deeper ResNet architectures. We experimented with ResNet (He et al., 2015) architectures on **CIFAR-10** and **SVHN**, using the architectures listed in Table 8, that replicate the setup described in (Miyato et al., 2018) on **CIFAR-10**. For experiments with ResNet, we used the hinge version of the adversarial non-saturating loss, Eq. 70 & 71. See § F for details on the hyperparameters.

F. Additional Experiments

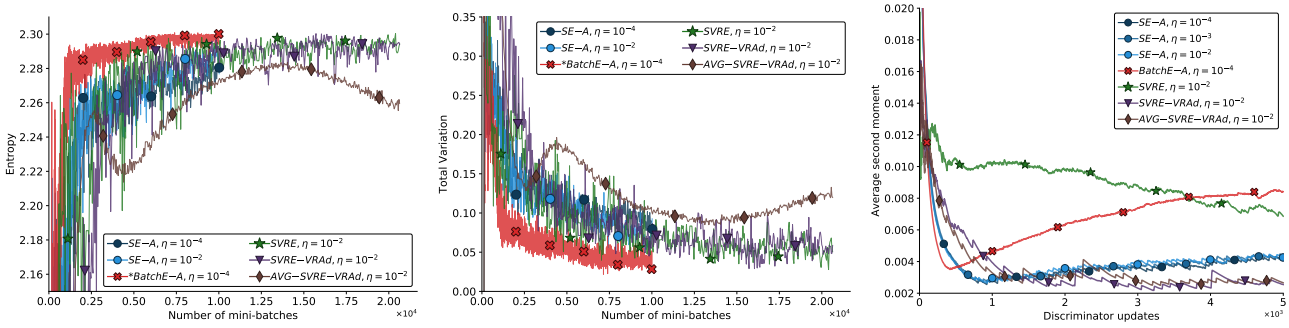
MNIST. The results in Table 2 on **MNIST** are obtained using 5 runs with different seeds, and the shown performances are the averaged values. Each experiment was run for $100K$ iterations. The corresponding scores with the standard deviations are as follows: (i) IS: $8.62\pm.02$, $8.58\pm.08$, $8.56\pm.11$; (ii) FID: $0.17\pm.03$, $0.15\pm.01$, $0.18\pm.02$; for SE–A, SVRE, and SVRE–VRAd, respectively. On this dataset, we obtain similar final performances if run for many iterations, however SVRE converges faster (see Fig. 3). Fig. 6 illustrates additional metrics of the experiments shown in Fig. 3.

Deeper ResNet architectures on CIFAR-10 and SVHN. For the SE–A baseline, we observe that the convergence is notably more unstable when using deeper architectures. In particular, either the training fails to converge or it diverges at later iterations. When updating G and D equal number of times *i.e.* using $1:1$ update ratio, on **CIFAR-10** we obtained best FID score of 24.91 using $\eta_G = 2 \times 10^{-4}$, $\eta_D = 4 \times 10^{-4}$, $\beta_1 = 0$, while experimenting with several combinations of η_G, η_D, β_1. Using exponential learning rate decay with a multiplicative factor of 0.99, improved the best FID score to 20.70, obtained for the experiment with $\eta_G = 2 \times 10^{-4}$, $\eta_D = 2 \times 10^{-4}$, $\beta_1 = 0$. Finally, using $1:5$ update ratio, with $\eta_G = 2 \times 10^{-4}$, $\eta_D = 2 \times 10^{-4}$, $\beta_1 = 0$ provided best FID of 18.65 for the baseline.

We observe that SVRE is more stable in terms of hyper-parameter selection, as it always starts to converge and does not diverge at later iterations. Compared to shallower iterations, we observe that SVRE takes longer to converge than its baseline for this architecture. With constant step size of $\eta_G = 1 \times 10^{-3}$, $\eta_D = 4 \times 10^{-3}$ we obtain FID score of 23.56 on **CIFAR-10**. Our results suggest that SVRE can be further combined with adaptive step size schemes, so as to obtain both stable GAN performances and fast convergence when using these architectures.

Generator	Discriminator
Input: $z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$	*Input:* $x \in \mathbb{R}^{3\times32\times32}$
Linear($128 \rightarrow 4096$)	D–ResBlock
G–ResBlock	D–ResBlock
G–ResBlock	D–ResBlock
G–ResBlock	D–ResBlock
Batch Normalization	ReLU
ReLU	AvgPool (kernel:8×8)
conv. (kernel: 3×3, $256 \rightarrow 3$; stride: 1; pad:1)	Linear($128 \rightarrow 1$)
$Tanh(\cdot)$	Spectral Normalization

Table 8: ResNet architectures used for experiments on **SVHN** and **CIFAR-10**, where G–ResBlock and D–ResBlock for the Generator and the Discriminator, respectively, are described in Table 7. The models' parameters are initialized using the Xavier initialization (Glorot and Bengio, 2010).



(a) Entropy (higher is better) (b) Total variation (lower is better) (c) Discriminator

Figure 6: Stochastic, full-batch and variance reduced versions of the extragradient method ran on **MNIST**, see § 5.1. **BatchE–A* emphasizes that this method is **not** scaled with the number of passes (x-axis). The input space is $1\times28\times28$, see § E.2 for details on the implementation.