Training Generative Adversarial Networks via Stochastic Nash Games

Franci, Barbara; Grammatico, Sergio

DOI
10.1109/TNNLS.2021.3105227

Publication date
2023

Document Version
Final published version

Published in
IEEE Transactions on Neural Networks and Learning Systems

Training Generative Adversarial Networks via Stochastic Nash Games

Barbara Franci and Sergio Grammatico, *Senior Member, IEEE*

***Abstract*— Generative adversarial networks (GANs) are a class of generative models with two antagonistic neural networks: a generator and a discriminator. These two neural networks compete against each other through an adversarial process that can be modeled as a stochastic Nash equilibrium problem. Since the associated training process is challenging, it is fundamental to design reliable algorithms to compute an equilibrium. In this article, we propose a stochastic relaxed forward-backward (SRFB) algorithm for GANs, and we show convergence to an exact solution when an increasing number of data is available. We also show convergence of an averaged variant of the SRFB algorithm to a neighborhood of the solution when only a few samples are available. In both cases, convergence is guaranteed when the pseudogradient mapping of the game is monotone. This assumption is among the weakest known in the literature. Moreover, we apply our algorithm to the image generation problem.**

***Index Terms*— Generative adversarial networks (GANs), stochastic Nash equilibrium (SNE) problems (SNEPs), two-player game, variational inequalities.**

I. Introduction

A. Generative Adversarial Networks

GENERATIVE adversarial networks (GANs) is an example of an unsupervised generative model. The basic idea is that, given some samples drawn from a probability distribution, the neural network takes a training set and learns how to obtain an estimate of such distribution. Most of the literature on GANs focuses on sample generation (especially image generation), but they can also be designed to explicitly estimate a probability distribution [1]–[4].

The learning process of the neural networks in GANs is made via an adversarial process, in which not only the generative model but also the opponent are simultaneously trained. Indeed, there are two neural network classes: the generator that creates data according to a given distribution, and the discriminator that tries to recognize if the samples come from the training data or the generator. As an example, the generator can be considered as a team of counterfeiters, trying to produce fake currency, while the discriminative model, i.e., the police, tries to detect the counterfeit money [2]. To succeed in this game, the former must learn to reproduce money that is indistinguishable from the original currency, while the discriminator must recognize the samples that are drawn from the same distribution as the training data. Through the competition, both teams improve their methods until the counterfeit currency is indistinguishable from the original.

Besides this simplistic interpretation, the subject has been widely studied in the literature, because it has many and various applications. In addition to the classic image generation problem [1], [5], GANs have been applied in medicine, e.g., to improve the diagnostic performance of the low-dose computed tomography method [6] and recently to detect pneumonia in potential Covid-19 patients [7]. Moreover, they can be used for correcting images taken under adverse weather conditions, as rain [8], [9] or fog [10], editing facial attributes [11], image inpainting [12], [13] as well as Pacman [14].

B. Stochastic Nash Equilibrium Problems

The reason why these networks are called *adversarial* is related to the fact that they can be modeled as a game, where each agent payoff depends on the variables of the other agent [15], [16]. However, the players in GANs can also be considered to be cooperative since they share information with each other [1], [17]. Since there are only the generator and the discriminator, the problem is an instance of a two-player game and it can be also cast as a zero-sum game, depending on the choice of the cost functions. From a more general point of view, the class of games that suits the GAN problem is that of stochastic Nash equilibrium (SNE) problems (SNEPs) where each agent tries to minimize its expected value cost function. Given their connection with game theory, GANs have received theoretical attention as well, both on the study of the associated Nash equilibrium problem [16], [18] and on the design of algorithms to improve the learning process [18], [19].

Among the available methods to solve a SNEP, an elegant approach is to recast the problem as a stochastic variational inequality (SVI) [19]–[21]. The advantage of this approach is that there are many algorithms available for finding a solution of an SVI, some of them already applied to GANs [19], [22]. For instance, the most used in machine learning is the forward-backward (FB) algorithm [23], also known as gradient descent [24], which has the disadvantage that, to ensure convergence, the mapping should be cocoercive, i.e., strongly monotone and Lipschitz continuous. Since the GAN mapping is often nonconvex [17], [19], one would prefer an algorithm that is guaranteed to converge for at most monotone mappings. In this case, one may consider the extragradient (EG) algorithm [25]–[27] and the forward-backward-forward (FBF) algorithm [28] which, however, require two costly evaluations

Manuscript received 18 November 2020; revised 10 May 2021 and 22 July 2021; accepted 12 August 2021. Date of publication 26 August 2021; date of current version 1 March 2023. This work was supported in part by NWO through Research Projects OMEGA and P2P-TALES under Grant 613.001.702 and Grant 647.003.003 and in part by the ERC through Research Project COSMOS under Grant 802348. *(Corresponding author: Barbara Franci.)*

The authors are with the Delft Center for System and Control, TU Delft, Delft 2600AA, The Netherlands (e-mail: b.franci-1@tudelft.nl; s.grammatico@tudelft.nl).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3105227.

Digital Object Identifier 10.1109/TNNLS.2021.3105227

of the pseudogradient mapping, i.e., they are computationally expensive. Due to the large-scale problem size, the ideal algorithm should not be computationally demanding and it should be guaranteed to converge under nonrestrictive assumption on the pseudogradient mapping.

C. Contribution

Motivated by the need for computationally light iterations converging under weak assumptions, we propose an algorithm that requests only one computation of the pseudogradient mapping per iteration and we show its convergence under mere monotonicity. Specifically, our contributions are the following.

1) We propose a stochastic relaxed FB (SRFB) algorithm and a variant with averaging (aSRFB) for the training process of GANs. The SRFB involves only one evaluation of the pseudogradient mapping at each iteration, therefore it is computationally cheaper than the EG and FBF algorithms.
2) We prove its convergence for monotone mappings, which is considered the "weakest possible" assumption on the pseudogradient mapping [29]. Specifically, whenever only a finite number of samples are available, we prove almost sure convergence to a neighborhood of the solution, while if an increasing set of samples is available, then the algorithm reaches an equilibrium almost surely.
3) We apply our algorithm to the image generation problem and compare it with the EG scheme.

Our SRFB algorithm is inspired by the works [30] and [31], and a preliminary heuristic application to GAN was presented in [32]. Therein, we do not prove convergence of the SRFB algorithm nor of its aSRFB variant. Moreover, in [32], we only run numerical simulations on synthetic toy examples while in this article we train the two neural networks for the popular image generation problem with real benchmark data.

D. Related Work

Due to the connection between SNEPs and SVIs, many algorithms for variational inequalities have been applied to GANs [19], [21]. The first one is the FB algorithm [23], [33], also known as gradient descent [24]. It is the most used, even if in many cases it has been proven to be nonconvergent [19], [34]. From an operator-theoretic perspective, the FB is not convergent because the pseudogradient mapping should be cocoercive [35] and this is almost never the case in GANs. From an algorithmic perspective, the iterates typically cycle in a neighborhood of a solution without reaching it [34]. Therefore, research has focused on the FBF algorithm and on the EG algorithm that are guaranteed to converge for merely monotone mappings. The FBF algorithm, first presented in [36] and extended to the stochastic case in [28], involves two evaluations of the pseudogradient mapping. A first attempt to apply the FBF algorithm for GANs is presented, along with a relaxed inertial FBF algorithm, in [37]. The EG method was first proposed in [38] and extended many years later to the stochastic case in [25] and [26] and to GANs in [19] and [27]. The EG algorithm requires two evaluations of the pseudogradient mapping as well, therefore in [19], a variation is proposed. This involves an extrapolation from the past, i.e., it uses the evaluation of the mapping at previous time steps. Gidel *et al.* [19] propose also the FB and the EG algorithms with averaging.

The averaging technique was first proposed for VIs in [24] and studied more recently in [19] and [39]. Yazici *et al.* [39] examined two different techniques for averaging: the moving average, which computes the time-average of the iterates, and the exponential moving average which computes an exponentially discounted sum. For both the techniques, they show that despite convergence cannot be proven, the averaging may help stabilizing the iterates, driving them toward a neighborhood of the solution. While [39] has mostly a heuristic approach, theoretical convergence studies are presented in [34] and [40]. Therein, the authors show that the local convergence and stability properties of GAN training depend on the eigenvalues of the Jacobian of the associated gradient vector field.

Another theoretical aspect that has not been extensively addressed yet is the inherent relation between GANs and game theory. Oliehoek *et al*. [16] formally introduce GAN Games, describing (and seeking for) the Nash equilibria of the zero-sum game as saddle points in mixed strategies. The study of saddle point problems is also studied, in connection with GANs, in [22]. Heusel *et al*. [17], instead, prove that Adam [41], a second-order method for GANs, converges to a stationary local Nash equilibrium.

E. Notation

Let $\mathbb{R}$ indicate the set of real numbers and let $\bar{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$. $\langle\cdot,\cdot\rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ denotes the standard inner product and $\|\cdot\|$ represents the associated Euclidean norm. Given N vectors $x_1, \ldots, x_N \in \mathbb{R}^n$, $\boldsymbol{x} := \mathrm{col}(x_1, \ldots, x_N) = \left[x_1^\top, \ldots, x_N^\top\right]^\top$. For a closed set $C \subseteq \mathbb{R}^n$, the mapping $\mathrm{proj}_C : \mathbb{R}^n \to C$ denotes the projection onto C, i.e., $\mathrm{proj}_C(x) = \mathrm{argmin}_{y\in C} \|y - x\|$.

II. Generative Adversarial Networks

The idea behind GANs is to set up an antagonistic training process between the generator and the discriminator. Typically, the generator and the discriminator are represented by two deep neural networks, and accordingly, they are denoted by two functions, differentiable with respect to their inputs and parameters. The generator creates samples that aim at resembling the distribution of the training data. hence, it is trained to fool the discriminator who, in turn, examines the samples to determine whether they are real or fake. This adversarial mechanism can be modeled as a game where the generator and the discriminator represent the players, who want to improve their *payoff* [16].

Formally, the generator is a neural network class, represented by a differentiable function g, with parameters vector $x_g \in \boldsymbol{\Omega}_g \subseteq \mathbb{R}^{n_g}$. Let us denote the (fake) output of the generator with $g(z, x_g) \in \mathbb{R}^q$ where the input z is a random noise vector drawn from the data prior distribution, $z \sim p_z$ [16]. In game-theoretic terms, the *strategies* of the generator are the parameters x_g that allow g to generate the fake output.

Similarly, the discriminator is a neural network class with parameter vector $x_d \in \boldsymbol{\Omega}_d \subseteq \mathbb{R}^{n_d}$ and a single output $d(v, x_d) \in [0, 1]$ that indicates how good is the input v. The output of the discriminator can be interpreted as the probability of being real that d assigns to an element v. The *strategies* of the discriminator are the parameters x_d. Usually [2], [19], the payoff of the discriminator is given by the function

$$J_d(x_g, x_d) = \mathbb{E}[\phi(d(\cdot, x_d))] - \mathbb{E}[\phi(d(g(\cdot, x_g), x_d))] \quad (1)$$

where $\phi : [0, 1] \to \mathbb{R}$ is a measuring function. The typical choices for ϕ are the Kullback-Leibler divergence or the Jensen-Shannon divergence (a logarithm) as in [2] but other options (such as the Wasserstein distance) are proposed in the literature [3], [42]. Regardless, the mapping in (1) can be interpreted as the distance between the fake value and the real one. The payoff of the generator (J_g) instead, depends on how we describe the game. In fact, the problem can be modeled as a two-player game, or as a zero-sum game, depending on the cost functions. To cast the problem as a zero-sum game, the functions J_g and J_d should satisfy the following relation:

$$J_g(x_g, x_d) = -J_d(x_g, x_d). \quad (2)$$

Then, we can rewrite it as a minmax problem, that is,

$$\min_{x_g} \max_{x_d} J_d(x_g, x_d). \quad (3)$$

In other words, (3) means that the generator aims at minimizing the distance between the real value and the fake one, while the discriminator wants to maximize such a distance, i.e., d aims at recognizing the generated data. When the generator has a different payoff function from the discriminator, e.g., [19]

$$J_g(x_g, x_d) = \mathbb{E}[\phi(d(g(\cdot, x_g), x_d))], \quad (4)$$

then the problem is not a zero-sum game.

Since the two-player game with cost functions (1) and (4) and the zero-sum game with cost function (1) and relation (2) have the same pseudogradient mapping (defined Section III), it can be proven that the two equilibria are strategically equivalent [16, Th. 10].

III. Stochastic Nash Equilibrium Problems

In this section, let us describe the GAN game as a generic SNEP. The two neural network classes are indexed by the set $\mathcal{I} = \{g, d\}$. Each agent $i \in \mathcal{I}$ has a decision variable $x_i \in \boldsymbol{\Omega}_i \subseteq \mathbb{R}^{n_i}$. In general, the local cost function of agent $i \in \mathcal{I}$ is defined as

$$\mathbb{J}_i(x_i, x_j) = \mathbb{E}_\xi[J_i(x_i, x_j, \xi(\omega))] \quad (5)$$

for some measurable function $J_i : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ where $n = n_d + n_g$. The cost function $\mathbb{J}_i$ of agent $i \in \mathcal{I}$ depends on its local variable x_i, the decisions of the other player x_j, $j \neq i$, and the random variable $\xi : \Xi \to \mathbb{R}^d$ that represent the uncertainty. The latter arises when we do not know the distribution of the random variable or it is computationally too expansive to compute the expected value. In practice, this means that we have access only to a finite number of samples from the data distribution. Given the probability space $(\Xi, \mathcal{F}, \mathbb{P})$, $\mathbb{E}_\xi$ indicates the mathematical expectation with respect to the distribution of the random variable $\xi(\omega)$[1]; $\mathbb{E}[J_i(\boldsymbol{x}, \xi)]$ is well defined for all the feasible $\boldsymbol{x} = \mathrm{col}(x_g, x_d) \in \boldsymbol{\Omega} = \boldsymbol{\Omega}_g \times \boldsymbol{\Omega}_d$. For our theoretical analysis, some assumptions on the cost function and the feasible set should be postulated. The following assumptions are standard in monotone game theory [43], [44].

Assumption 1: For each $i \in \mathcal{I}$, the set $\boldsymbol{\Omega}_i$ is nonempty, compact, and convex.

For each $i, j \in \mathcal{I}$, $i \neq j$, the function $\mathbb{J}_i(\cdot, x_j)$ is convex and continuously differentiable. For each $i \in \mathcal{I}$, $j \neq i$ and for each $\xi \in \Xi$, the function $J_i(\cdot, x_j, \xi)$ is convex, continuously differentiable and Lipschitz continuous with the constant $\ell_i(x_j, \xi)$ integrable in ξ. The function $J_i(x_i, x_j, \cdot)$ is measurable for each x_j, $j \neq i$. ■

Given the decision variable of the other agent, the aim of each agent i is to choose a strategy x_i that solves its local optimization problem, that is

$$\forall i \in \mathcal{I} : \min_{x_i \in \boldsymbol{\Omega}_i} \mathbb{J}_i(x_i, x_j). \quad (6)$$

The solution of the coupled optimization problems in (6) that we are seeking is an SNE [44].

Definition 1: An SNE is a collective strategy $\boldsymbol{x}^* = \mathrm{col}(x_g^*, x_d^*) \in \boldsymbol{\Omega}$ such that for all $i \in \mathcal{I}$

$$\mathbb{J}_i(x_i^*, x_j^*) \leq \inf\{\mathbb{J}_i(y, x_j^*) \mid y \in \boldsymbol{\Omega}_i\}.$$

In other words, an SNE is a pair of strategies where neither the generator nor the discriminator can decrease its cost function by unilaterally deviating from its decision.

Although existence of an SNE of the game in (6) is guaranteed, under Assumption 1 [44, Sect. 3.1], uniqueness does not hold in general [44, Sect. 3.2].

To seek for a Nash equilibrium, we rewrite the problem as a SVI. Let us first denote the pseudogradient mapping as

$$\mathbb{F}(\boldsymbol{x}) = \begin{bmatrix} \mathbb{E}[\nabla_{x_g} J_g(x_g, x_d)] \\ \mathbb{E}[\nabla_{x_d} J_d(x_d, x_g)] \end{bmatrix}. \quad (7)$$

We note that the possibility to exchange the expected value and the pseudogradient is ensured by Assumption 1 [44].

Then, the associated SVI reads as

$$\langle \mathbb{F}(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle \geq 0 \quad \text{for all} \quad \boldsymbol{x} \in \boldsymbol{\Omega}. \quad (8)$$

Remark 1: If Assumption 1 holds, then $\boldsymbol{x}^* \in \boldsymbol{\Omega}$ is a Nash equilibrium of the game in (6) if and only if $\boldsymbol{x}^*$ is a solution of the SVI in (8) [20, Prop. 1.4.2], [44, Lemma 3.3].

Moreover, under Assumption 1, the solution set of $\mathrm{SVI}(\boldsymbol{\Omega}, \mathbb{F})$ is nonempty and compact, i.e., $\mathrm{SOL}(\boldsymbol{\Omega}, \mathbb{F}) \neq \varnothing$ [20, Corollary 2.2.5] and an equilibrium exists. ■

In light of Remark 1, we call variational equilibria (v-SNE) the solution of the $\mathrm{SVI}(\boldsymbol{\Omega}, \mathbb{F})$ in (8) where $\mathbb{F}$ is as in (7), i.e., the solution of the SVI that are also SNE.

IV. Stochastic Relaxed Forward-Backward Algorithms

In this section, we propose two algorithms for solving the SNEP associated with the GANs process: a stochastic relaxed forward backward (SRFB) algorithm and its variant with averaging (aSRFB). The iterations read as in Algorithm 1

[1]From now on, simplicity, we use ξ instead of $\xi(\omega)$ and $\mathbb{E}$ instead of $\mathbb{E}_\xi$.

Algorithm 1 SRFB

1 Initialization: $x_i^0 \in \Omega_i$

2 Iteration k: Agent i receives x_j^k for $j \neq i$, then updates:

$$\bar{x}_i^k = (1-\delta)x_i^k + \delta\bar{x}_i^{k-1} \tag{11a}$$
$$x_i^{k+1} = \mathrm{proj}_{\Omega_i}\left[\bar{x}_i^k - \lambda_i F_i^{\mathrm{VR}}\left(x_i^k, x_j^k, \xi_i^k\right)\right] \tag{11b}$$

and Algorithm 2, respectively, and they represent the steps for each agent $i \in \{g, d\}$.

Algorithms 1 and 2 differ, besides the presence of the averaging step, on the choice of the approximation used for the pseudogradient mapping. Moreover, we note that the averaging step in Algorithm 2, namely,

$$\boldsymbol{X}^K = \frac{\sum_{k=1}^K \lambda_k \boldsymbol{x}^k}{S_K}, \quad S_K = \sum_{k=1}^K \lambda_k \tag{9}$$

can be implemented in a first-order fashion as

$$\boldsymbol{X}_K = \left(1-\tilde{\lambda}_K\right)\boldsymbol{X}^{K-1} + \tilde{\lambda}_K \boldsymbol{x}^K \tag{10}$$

for some $\tilde{\lambda}_K \in [0,1]$. Moreover, let us remark that (10) is different from (11a) and (14a). Indeed, in Algorithms 1 and 2, (11a) and (14a) are convex combinations, with a constant parameter δ, of the two previous iterates $\boldsymbol{x}^k$ and $\bar{\boldsymbol{x}}^{k-1}$, while the averaging in (10) is a weighted cumulative sum over the decision variables $\boldsymbol{x}^k$ for all the iterations $k \in \{1,\ldots,K\}$, with time–varying weights $\{\tilde{\lambda}_k\}_{k=1}^K$. The parameter $\tilde{\lambda}_K$ can be tuned to obtain uniform, geometric, or exponential averaging [19], [39].

Let us now describe the approximation schemes used in the definitions of the algorithms. In the SVI framework, there are two main possibilities, depending on the samples available.

Using a finite, fixed number of samples is called stochastic approximation (SA) [23], and it is widely used in the literature of SVIs, in conjunction with conditions on the step sizes to control the stochastic error [26], [45]. In fact, unless the step size sequence is diminishing, it is only possible to prove convergence to a neighborhood of a solution. The SA of the pseudogradient mapping, given one sample of the random variable reads as

$$F^{\mathrm{SA}}(\boldsymbol{x},\xi) := \begin{bmatrix} \nabla_{x_g} J_g(x_g, x_d, \xi_g) \\ \nabla_{x_d} J_d(x_d, x_g, \xi_d) \end{bmatrix}. \tag{12}$$

F^{SA} uses one or a finite number, called minibatch, of realizations of the random variable.

When a huge number of samples is available, one can consider using a different approximation scheme, that is,

$$F^{\mathrm{VR}}(\boldsymbol{x},\xi^k) = \begin{bmatrix} \frac{1}{N_k}\sum_{s=1}^{N_k} \nabla_{x_g} J_i(x_g^k, x_d^k, \xi_g^{(s)}) \\ \frac{1}{N_k}\sum_{s=1}^{N_k} \nabla_{x_d} J_i(x_d^k, x_g^k, \xi_d^{(s)}) \end{bmatrix}. \tag{13}$$

In this case, an increasing number of samples, the batch size N_k, is taken at each iteration [25]. The superscript VR stands for variance reduction and it is related to the property of the approximation error discussed in Remark 2.

Algorithm 2 SRFB With Averaging

1 Initialization: $x_i^0 \in \Omega_i$

2 Iteration $k \in \{1,\ldots,K\}$: Agent i receives x_j^k for $j \neq i$, then updates:

$$\bar{x}_i^k = (1-\delta)x_i^k + \delta\bar{x}_i^{k-1} \tag{14a}$$
$$x_i^{k+1} = \mathrm{proj}_{\Omega_i}[\bar{x}_i^k - \lambda_i F_i^{\mathrm{SA}}(x_i^k, x_j^k, \xi_i^k)] \tag{14b}$$

3 Iteration K: $X_i^K = \frac{\sum_{k=1}^K \lambda_k x_i^k}{\sum_{k=1}^K \lambda_k}$

V. Convergence Analysis

A. Basic Technical Assumptions

With the aim of proving convergence to a solution (or to its neighborhood) of Algorithms 1 and 2, we start this section with the assumptions that are common to both algorithms.

The following monotonicity assumption on the pseudogradient mapping is standard for SVI problems [25], [31], also when applied to GANs [19] and it is the weakest possible to hope for global convergence.

Assumption 2: $\mathbb{F}$ in (7) is monotone, i.e., $\langle \mathbb{F}(\boldsymbol{x}) - \mathbb{F}(\boldsymbol{y}), \boldsymbol{x}-\boldsymbol{y}\rangle \geq 0$ for all $\boldsymbol{x},\boldsymbol{y} \in \boldsymbol{\Omega}$. ■

Let us now define the filtration $\mathcal{F} = \{\mathcal{F}_k\}$, that is, a family of σ-algebras such that $\mathcal{F}_0 = \sigma(X_0)$, $\mathcal{F}_k = \sigma(X_0, \xi_1, \xi_2, \ldots, \xi_k)$ for all $k \geq 1$, and $\mathcal{F}_k \subseteq \mathcal{F}_{k+1}$ for all $k \geq 0$. For all $k \geq 0$, let us also define the stochastic error as

$$\epsilon_k = \hat{F}(\boldsymbol{x}^k, \xi^k) - \mathbb{F}(\boldsymbol{x}^k) \tag{15}$$

where $\hat{F}$ indicates one of the two possible approximation schemes. In words, ϵ_k in (15) is the distance between the approximation and the exact expected value mapping. Then, let us postulate that the stochastic error has zero mean and bounded variance, as usual in SVI [19], [25], [31].

Assumption 3: The stochastic error is such that, for all $k \geq 0$, a.s., $\mathbb{E}[\epsilon^k|\mathcal{F}_k] = 0$ and $\mathbb{E}[\|\epsilon^k\|^2|\mathcal{F}_k] \leq \sigma^2$. ■

B. Convergence of Algorithm 1

We now state the convergence result for Algorithm 1. First, let us postulate some assumptions functional to our analysis. We start with the batch size sequence, which should be increasing to control the stochastic error.

Assumption 4: The batch size sequence $(N_k)_{k\geq 1}$ is such that, for some $b, k_0, a > 0$, $N_k \geq b(k+k_0)^{a+1}$. ■

Remark 2: Given F^{VR} as in (13), it can be proven that, for some $C > 0$

$$\mathbb{E}\left[\|\epsilon_k\|^2|\mathcal{F}_k\right] \leq \frac{C\sigma^2}{N_k}$$

i.e., the error diminishes as the batch size increases. Such result is, therefore, called variance reduction. More details can be found in [25, Lemma 3.12] and [33, Lemma 6]. ■

In addition to Assumption 2, we postulate that the pseudogradient mapping is Lipschitz continuous.

Assumption 5: $\mathbb{F}$ in (7) is ℓ-Lipschitz continuous for $\ell > 0$, i.e., $\|\mathbb{F}(\boldsymbol{x}) - \mathbb{F}(\boldsymbol{y})\| \leq \ell\|\boldsymbol{x}-\boldsymbol{y}\|$ for all $\boldsymbol{x},\boldsymbol{y} \in \boldsymbol{\Omega}$. ■

Using the variance reduced scheme in (13), we can take a constant step size, as long as it is small enough while the relaxation parameter should not be too small.

Assumption 6: The step size in Algorithm 1 is such that $\lambda \in (0, 1/(2\delta(2\ell+1))]$ where ℓ is the Lipschitz constant of $\mathbb{F}$ in (7) as in Assumption 5. The relaxation parameter in Algorithm 1 is such that $\delta \in \left[2/(1+\sqrt{5}), 1\right]$. ■

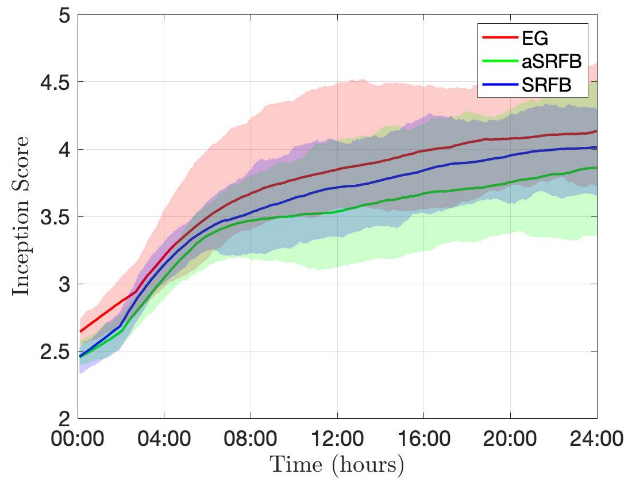

Fig. 1. Inception scores reached by the EG, the SRFB and the aSRFB algorithms.

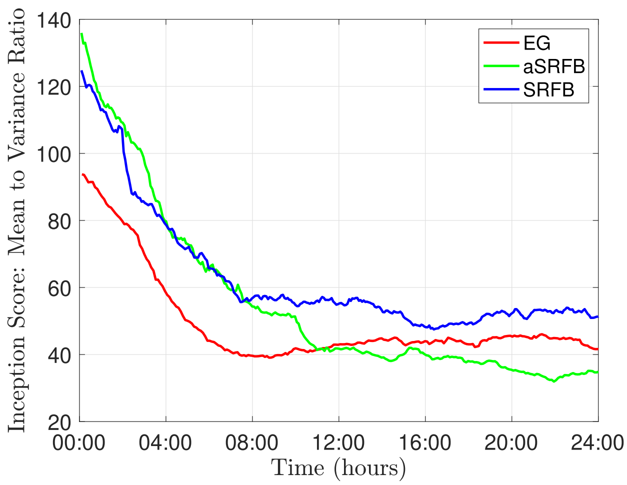

Fig. 2. Mean to variance ratio corresponding to the average inception scores.

We can finally state our first convergence result.

Theorem 1: Let Assumptions 1–6 hold. Then, the sequence $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ generated by Algorithm 1 with F^{VR} as in (13) converges a.s. to a SNE of the game in (6).

Proof: See Appendix B. □

C. Convergence of Algorithm 2

In this section, we state the convergence result (and the required assumptions) for Algorithm 2.

First, the bound on the relaxation parameter is wider in this case (compared to Assumption 6).

Assumption 7: The relaxation parameter in Algorithm 2 is such that $\delta \in (0, 1)$. ■

Next, we postulate an assumption on the SA approximation in (12), reasonable in our game-theoretic framework [19]. We also assume an explicit bound on the feasible set.

Assumption 8: F^{SA} in (12) is bounded, i.e., there exists $B > 0$ such that for $\boldsymbol{x} \in \boldsymbol{\Omega}$, $\mathbb{E}[\|F^{\mathrm{SA}}(\boldsymbol{x}, \xi)\|^2 | \mathcal{F}_k] \leq B$. ■

Assumption 9: The local constraint set $\boldsymbol{\Omega}$ is such that $\max_{\boldsymbol{x},\boldsymbol{y}\in\boldsymbol{\Omega}} \|\boldsymbol{x} - \boldsymbol{y}\|^2 \leq R^2$, for some $R \geq 0$. ■

To measure how close a point is to the solution, let us introduce the gap function

$$\mathrm{err}(\boldsymbol{x}) = \max_{\boldsymbol{x}^*\in\boldsymbol{\Omega}} \langle \mathbb{F}(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^* \rangle \tag{16}$$

TABLE I

NEURAL NETWORKS USED

Generator
Input $z \in \mathbb{R}^{1\times 1\times 100}$
Linear $100 \to 5 \times 5 \times 512$
Transp. Conv (ker: 5×5, $512 \to 256$, stride: 1, crop: 0)
Batch Normalization
ReLu
Transp. Conv (ker: 5×5, $256 \to 128$, stride: 2, crop: 1)
Batch Normalization
ReLu
Transp. Conv (ker: 5×5, $128 \to 3$, stride: 2, crop: 1)
Tanh($\cdot$)

Discriminator
Input $z \in \mathbb{R}^{64\times 64\times 3}$
Conv (ker: 5×5, $3 \to 64$, stride: 2, pad: 1)
LeakyReLu
Conv (ker: 5×5, $64 \to 256$, stride: 2, pad: 1)
Batch Normalization
LeakyReLu
Conv (ker: 5×5, $256 \to 512$, stride: 2, pad: 1)
Batch Normalization
LeakyReLu
Conv (ker: 5×5, $512 \to 1$)

which is equal 0 if and only if $\boldsymbol{x}$ is a solution of the (S)VI in (8) [20, eq. (1.5.2)]. Other possible measure functions can be found in [19].

We are now ready to state our second convergence result.

Theorem 2: Let Assumptions 1–3 and 7–9 hold. Let $\boldsymbol{X}^K = (1/K)\sum_{k=1}^K \boldsymbol{x}^k$, $c = (2 - \delta^2/1 - \delta)$, B be as in Assumption 8, R be as in Assumption 9 and σ^2 be as in Assumption 3. Then, the sequence $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ generated by Algorithm 2 with constant step size and F^{SA} as in (12) is such that

$$\mathbb{E}[\mathrm{err}(\boldsymbol{X}^K)] = \tfrac{cR}{\lambda K} + (2B^2 + \sigma^2)\lambda.$$

Thus, $\lim_{K\to\infty} \mathbb{E}[\mathrm{err}(\boldsymbol{X}^K)] = (2B^2 + \sigma^2)\lambda$.

Proof: See Appendix C. □

Remark 3: The average defined in Theorem 2 is not in conflict with the definition in (9) because if we consider a fixed step size, it holds that

$$\boldsymbol{X}^K = \frac{\sum_{k=1}^K \lambda_k \boldsymbol{x}^k}{\sum_{k=1}^K \lambda_k} = \frac{\lambda \sum_{k=1}^K \boldsymbol{x}^k}{K\lambda} = \frac{1}{K}\sum_{k=1}^K \boldsymbol{x}^k. \quad ■$$

VI. NUMERICAL SIMULATIONS

Let us present some numerical experiments to validate the analysis. We show how GANs are trained using our SRFB algorithm and we propose a comparison with one of the most used algorithms for GANs. Specifically, we compare our SRFB algorithm with the EG algorithm (Algorithm 3) [19]. We note that, compared to Algorithm 1, Algorithm 3 involves two projection steps and two evaluations of the pseudogradient mapping. For the simulations we use Adam (Algorithm 4) [41] instead of the stochastic gradient [23]. In Algorithm 5 we propose the Relaxed Adam, i.e., the SRFB algorithm with Adam; the EG algorithm with Adam can be derived similarly [19, Algorithm 4]. All the simulations are performed on MATLAB R2020a with 128 G RAM and 2 * Intel(R) Xeon(R) Gold 6148 CPU at 2.40 GHz (20 cores each).

(a) (b) (c)

Fig. 3. Generated images with (a) SRFB algorithm, (b) aSFRB algorithm, and (c) EG algorithm.

We train two DCGAN architectures [3], [46] (presented in Table I) on the CIFAR10 dataset [47] with the GAN objective [1], [2]. We choose the hyperparameters of Adam as

Algorithm 3 EG

1 Initialization: $x_i^0 \in \Omega_i$

2 Iteration k: Agent i

3 Receives x_j^k for $j \neq i$, then updates:

$$y_i^k = \mathrm{proj}_{\Omega_i}[x^k - \alpha_k \hat{F}_i(x_i^k, x_j^k, \xi_i^k)]$$

Receives y_j^k for $j \neq i$, then updates:

$$x_i^{k+1} = \mathrm{proj}_{\Omega_i}[x^k - \alpha_k \hat{F}_i(y_i^k, y_j^k, \xi_i^k)]$$

Algorithm 4 Adam

Input : Initial parameters $x^0, \bar{x}^0 \in \boldsymbol{\Omega}$
Exponential decay rates $\beta_1, \beta_2 \in [0, 1)$
Step size α

Output: Parameters x^{k+1}

Initialize
- 1[st] moment vector $z^0 = 0$
- 2[nd] moment vector $y^0 = 0$
- Time step k=0

for $k = 1, \ldots, K$ **do**
- $\hat{g}^k = \hat{F}(x^k, \xi^k)$ # update pseudogradient
- $z^k = \beta_1 z^{k-1} + (1 - \beta_1)\hat{g}^k$ # update 1[st] moment
- $y^k = \beta_2 y^{k-1} + (1 - \beta_2)(\hat{g}^k)^2$ # update 2[nd] moment
- $\tilde{z}^k = \frac{z^k}{1-\beta_1^k}$ # compute 1[st] moment estimate
- $\tilde{y}^k = \frac{y^k}{1-\beta_2^k}$ # compute 2[nd] moment estimate
- $x^{k+1} = x^k - \alpha \frac{\tilde{z}^k}{\sqrt{\tilde{y}^k + \epsilon}}$ # update parameters

end

return x^{K+1}

$\beta_1 = 0.5$ and $\beta_2 = 0.9$. We compute the inception score [48] to have an objective comparison: the higher the inception score, the better the image generation. In Fig. 1, we show how the inception score increases with time; the solid lines represent a tracking average over the previous and following 50 values of the inceptions score, which is averaged over 20 runs. The transparent area indicated the maximum and minimum values obtained in the 20 runs.

We note that the SRFB algorithm is computationally less demanding than the EG algorithm. Specifically, in Fig. 1, after 24 h (86 400 s), the SRFB has performed approximately 13 0000 iterations while the EG 9 0000. The averaged aSRFB shows worse performances (after approximately 11 0000 iterations), but this is to be expected since we have convergence only to a neighborhood of the solution (Theorem 2). In Fig. 2, we show the mean to variance ratio (average Inception Score divided by its variance) at each time instant of the three algorithms. As one can see, from Fig. 1 and 2 the SRFB algorithm has a similar performance to the EG algorithm but with a smaller variance (higher ratio). This means that a new instance of the SRFB algorithm should be closer to the average performance than the EG, hence our results are more consistent. Fig. 3 shows a sample of the images generated by the algorithms.

Algorithm 5 Relaxed Adam

Input : Initial parameters $x^0, \bar{x}^0 \in \boldsymbol{\Omega}$
Exponential decay rates $\beta_1, \beta_2 \in [0, 1)$
Step size α
Relaxing parameter $\delta \in [\frac{2}{1+\sqrt{5}}, 1]$
Output: Parameters x^{k+1}

```
Initialize
    1st moment vector z^0 = 0
    2nd moment vector y^0 = 0
    Time step k = 0
for k = 1, ..., K do
    ĝ^k = F̂(x^k, ξ^k)   # update pseudogradient
    x̄^k = (1 − δ)x^k + δx̄^{k−1}   # relaxation step
    z^k = β1 z^{k−1} + (1 − β1)ĝ^k   # update 1st moment
    y^k = β2 y^{k−1} + (1 − β2)(ĝ^k)^2   # update 2nd moment
    z̃^k = z^k / (1 − β1^k)   # compute 1st moment estimate
    ỹ^k = y^k / (1 − β2^k)   # compute 2nd moment estimate
    x^{k+1} = x̄^k − α z̃^k / sqrt(ỹ^k + ε)   # update parameters
end
return x^K
```

VII. Conclusion

The SRFB algorithm is a very promising algorithm for training GANs. If an increasing number of samples is available and the pseudogradient mapping of the game is monotone, convergence to the exact solution holds. Instead, with only a finite, fixed minibatch and the same monotonicity assumption, convergence to a neighborhood of the solution can be proven by using an averaging technique. Our numerical experience shows a similar performance compared to the EG scheme, widely used in the literature for GANs.

In the future, it would be interesting to extend the convergence result to an exact solution also in the case of a small minibatch. Since the cost function associated with GAN is often nonconvex, it would also be worth finding algorithms converging under weaker assumptions than monotonicity.

Appendix A
Preliminary Results

We here recall some facts about norms and the projection operator and a preliminary result. Some results find inspiration from [30] where the algorithm is presented in the deterministic case. We start with the norms. We use the cosine rule

$$\langle x, y\rangle = \tfrac{1}{2}\big(\langle x, x\rangle + \langle y, y\rangle - \|x - y\|^2\big) \tag{17}$$

and the following two properties of the norm [49, Corollary 2.15], $\forall a, b \in \mathcal{E}$, $\forall \alpha \in \mathbb{R}$

$$\|\alpha a+(1-\alpha)b\|^2 = \alpha\|a\|^2 + (1-\alpha)\|b\|^2 - \alpha(1-\alpha)\|a-b\|^2 \tag{18}$$

$$\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2. \tag{19}$$

Regarding the projection operator, by [49, Prop. 12.26], it satisfies the following inequality: let C be a nonempty closed convex set, then, for all $x, y \in C$

$$\bar{x} = \operatorname{proj}_C(x) \Leftrightarrow \langle \bar{x} - x, y - \bar{x}\rangle \geq 0. \tag{20}$$

The projection is also firmly nonexpansive [49, Prop. 4.16], and consequently, quasi firmly nonexpansive [49, Def. 4.1].

The Robbins–Siegmund Lemma is widely used in literature to prove a.s. convergence of sequences of random variables.

Lemma 1 (Robbins-Siegmund Lemma, [50]): Let $\mathcal{F} = (\mathcal{F}_k)_{k\in\mathbb{N}}$ be a filtration. Let $\{\alpha_k\}_{k\in\mathbb{N}}$, $\{\theta_k\}_{k\in\mathbb{N}}$, $\{\eta_k\}_{k\in\mathbb{N}}$ and $\{\chi_k\}_{k\in\mathbb{N}}$ be nonnegative sequences such that $\sum_k \eta_k < \infty$, $\sum_k \chi_k < \infty$ and let $\forall k \in \mathbb{N}$, $\mathbb{E}[\alpha_{k+1}|\mathcal{F}_k]+\theta_k \leq (1+\chi_k)\alpha_k+\eta_k$ *a.s.* Then $\sum_k \theta_k < \infty$ and $\{\alpha_k\}_{k\in\mathbb{N}}$ converges a.s. to a nonnegative random variable. ■

The next lemma collects some properties that follow the definition of the SRFB algorithm.

Lemma 2: Given Algorithm 1, the following statements hold.

1) $\boldsymbol{x}^k - \bar{\boldsymbol{x}}^{k-1} = (1/\delta)(\boldsymbol{x}^k - \bar{\boldsymbol{x}}^k)$.
2) $\boldsymbol{x}^{k+1} - \boldsymbol{x}^* = (1/1-\delta)\big(\bar{\boldsymbol{x}}^{k+1} - \boldsymbol{x}^*\big) - (\delta/1-\delta)\big(\bar{\boldsymbol{x}}^k - \boldsymbol{x}^*\big)$.
3) $\delta/(1-\delta)^2\big\|\bar{\boldsymbol{x}}^{k+1} - \boldsymbol{x}^k\big\|^2 = \delta\big\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\big\|^2$.

Proof: Straightforward from Algorithm 1 and [30]. ■

Appendix B
Proof of Theorem 1

Proof of Theorem 1: Using the property of projection operator (20) we have

$$\big\langle \boldsymbol{x}^{k+1} - \bar{\boldsymbol{x}}^k + \lambda F^{\mathrm{VR}}(\boldsymbol{x}^k, \xi^k), \boldsymbol{x}^* - \boldsymbol{x}^{k+1}\big\rangle \geq 0 \tag{21}$$

$$\big\langle \boldsymbol{x}^k - \bar{\boldsymbol{x}}^{k-1} + \lambda F^{\mathrm{VR}}(\boldsymbol{x}^{k-1}, \xi^{k-1}), \boldsymbol{x}^{k+1} - \boldsymbol{x}^k\big\rangle \geq 0. \tag{22}$$

Using Lemma 2.1, (22) becomes

$$\big\langle \tfrac{1}{\delta}(\boldsymbol{x}^k - \bar{\boldsymbol{x}}^k) + \lambda F^{\mathrm{VR}}(\boldsymbol{x}^{k-1}, \xi^{k-1}), \boldsymbol{x}^{k+1} - \boldsymbol{x}^k\big\rangle \geq 0. \tag{23}$$

Then, adding (21) and (23) we obtain

$$\begin{aligned}&\big\langle \boldsymbol{x}^{k+1} - \bar{\boldsymbol{x}}^k + \lambda F^{\mathrm{VR}}(\boldsymbol{x}^k, \xi^k), \boldsymbol{x}^* - \boldsymbol{x}^{k+1}\big\rangle\\ &\quad+\big\langle \tfrac{1}{\delta}(\boldsymbol{x}^k - \bar{\boldsymbol{x}}^k) + \lambda F^{\mathrm{VR}}(\boldsymbol{x}^{k-1}, \xi^{k-1}), \boldsymbol{x}^{k+1} - \boldsymbol{x}^k\big\rangle \geq 0.\end{aligned} \tag{24}$$

Now we use the cosine rule (17)

$$\begin{aligned}&\big\langle \boldsymbol{x}^{k+1} - \bar{\boldsymbol{x}}^k, \boldsymbol{x}^* - \boldsymbol{x}^{k+1}\big\rangle\\ &\quad= -\tfrac{1}{2}\Big(\big\|\boldsymbol{x}^{k+1} - \bar{\boldsymbol{x}}^k\big\|^2 + \big\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^*\big\|^2 - \big\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}^k\big\|^2\Big)\\ &\quad\big\langle \tfrac{1}{\delta}(\boldsymbol{x}^k - \bar{\boldsymbol{x}}^k), \boldsymbol{x}^{k+1} - \boldsymbol{x}^k\big\rangle\\ &\quad= -\tfrac{1}{2\delta}\Big(\big\|\boldsymbol{x}^k - \bar{\boldsymbol{x}}^k\big\|^2 + \big\|\boldsymbol{x}^k - \boldsymbol{x}^{k+1}\big\|^2 - \big\|\boldsymbol{x}^{k+1} - \bar{\boldsymbol{x}}^k\big\|^2\Big)\end{aligned}$$

and we note that

$$\begin{aligned}&\lambda\big\langle F^{\mathrm{VR}}(\boldsymbol{x}^k, \xi^k), \boldsymbol{x}^* - \boldsymbol{x}^{k+1}\big\rangle\\ &\quad= -\lambda\big\langle \mathbb{F}(\boldsymbol{x}^k), \boldsymbol{x}^k - \boldsymbol{x}^*\big\rangle + \big\langle \epsilon^k, \boldsymbol{x}^* - \boldsymbol{x}^k\big\rangle\\ &\quad\quad+\lambda\big\langle \mathbb{F}(\boldsymbol{x}^k), \boldsymbol{x}^k - \boldsymbol{x}^{k+1}\big\rangle + \big\langle \epsilon^k, \boldsymbol{x}^k - \boldsymbol{x}^{k+1}\big\rangle.\end{aligned}$$

Then, by reordering and substituting in (24), we obtain

$$\begin{aligned}&-\big\|\boldsymbol{x}^{k+1} - \bar{\boldsymbol{x}}^k\big\|^2 - \big\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^*\big\|^2 + \big\|\boldsymbol{x}^* - \bar{\boldsymbol{x}}^k\big\|^2\\ &\quad+ - \tfrac{1}{\delta}\big\|\boldsymbol{x}^k - \bar{\boldsymbol{x}}^k\big\|^2 - \tfrac{1}{\delta}\big\|\boldsymbol{x}^k - \boldsymbol{x}^{k+1}\big\|^2 + \tfrac{1}{\delta}\big\|\boldsymbol{x}^{k+1} - \bar{\boldsymbol{x}}^k\big\|^2\\ &\quad+ - 2\lambda\big\langle \mathbb{F}(\boldsymbol{x}^k), \boldsymbol{x}^k - \boldsymbol{x}^*\big\rangle + 2\lambda\big\langle \varepsilon^k, \boldsymbol{x}^* - \boldsymbol{x}^k\big\rangle\\ &\quad+2\lambda\big\langle \mathbb{F}(\boldsymbol{x}^k) - \mathbb{F}(\boldsymbol{x}^{k-1}), \boldsymbol{x}^k - \boldsymbol{x}^{k+1}\big\rangle\\ &\quad+2\lambda\big\langle \varepsilon^k - \varepsilon^{k-1}, \boldsymbol{x}^k - \boldsymbol{x}^{k+1}\big\rangle \geq 0.\end{aligned} \tag{25}$$

Since $\mathbb{F}$ is monotone, it holds that $\langle \mathbb{F}(\boldsymbol{x}^k), \boldsymbol{x}^k - \boldsymbol{x}^*\rangle \geq \langle \mathbb{F}(\boldsymbol{x}^*), \boldsymbol{x}^k - \boldsymbol{x}^*\rangle \geq 0$. By using Lemma 2.2 and 2.3 as in (28), and substituting in (25), grouping and reordering, we get

$$\begin{aligned}&\tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^*\|^2+\tfrac{1}{\delta}\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2\\&\quad\le\left(\tfrac{\delta}{1-\delta}+1\right)\|\boldsymbol{x}^*-\bar{\boldsymbol{x}}^k\|^2-\tfrac{1}{\delta}\|\boldsymbol{x}^k-\bar{\boldsymbol{x}}^k\|^2\\&\quad\quad+2\lambda\langle\mathbb{F}(\boldsymbol{x}^k)-\mathbb{F}(\boldsymbol{x}^{k-1}),\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\rangle\\&\quad\quad+2\lambda\langle\varepsilon^k,\boldsymbol{x}^*-\boldsymbol{x}^k\rangle+2\lambda\langle\varepsilon^k-\varepsilon^{k-1},\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\rangle\end{aligned}\tag{26}$$

where we used Assumption 6. Moreover, by using Lipschitz continuity of $\mathbb{F}$ and Cauchy-Schwartz and Young's inequality, it follows that

$$\begin{aligned}&\lambda\langle\mathbb{F}(\boldsymbol{x}^k)-\mathbb{F}(\boldsymbol{x}^{k-1}),\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\rangle\\&\qquad\le\tfrac{\ell\lambda}{2}\left(\|\boldsymbol{x}^k-\boldsymbol{x}^{k-1}\|^2+\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2\right).\end{aligned}$$

Similarly, we can bound the term involving the stochastic errors

$$\begin{aligned}&2\lambda\langle\varepsilon^k-\varepsilon^{k-1},\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\rangle\\&\quad\le2\lambda\|\varepsilon^k-\varepsilon^{k-1}\|\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|\\&\quad\le\lambda\|\varepsilon^k-\varepsilon^{k-1}\|^2+\lambda\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2.\end{aligned}$$

By substituting in (26), we conclude that

$$\begin{aligned}&\tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^*\|^2+\tfrac{1}{\delta}\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2\\&\quad\le\tfrac{1}{1-\delta}\|\boldsymbol{x}^*-\bar{\boldsymbol{x}}^k\|^2-\tfrac{1}{\delta}\|\boldsymbol{x}^k-\bar{\boldsymbol{x}}^k\|^2\\&\quad\quad+\ell\lambda(\|\boldsymbol{x}^k-\boldsymbol{x}^{k-1}\|^2+\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2)\\&\quad\quad+\lambda\|\varepsilon^k-\varepsilon^{k-1}\|^2+\lambda\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2\\&\quad\quad+2\lambda\langle\varepsilon^k,\boldsymbol{x}^*-\boldsymbol{x}^k\rangle.\end{aligned}\tag{27}$$

Now, we consider the residual function of $\boldsymbol{x}^k$

$$\begin{aligned}\operatorname{res}(\boldsymbol{x}^k)^2&=\|\boldsymbol{x}^k-\operatorname{proj}(\boldsymbol{x}^k-\lambda\mathbb{F}(\boldsymbol{x}^k))\|^2\\&\le2\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2+2\|\bar{\boldsymbol{x}}_k-\boldsymbol{x}^k+\lambda\varepsilon_k\|^2\\&\le2\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2+4\|\bar{\boldsymbol{x}}_k-\boldsymbol{x}^k\|^2+4\lambda^2\|\varepsilon_k\|^2\end{aligned}$$

where we added and subtracted $\boldsymbol{x}^{k+1}=\operatorname{proj}(\bar{\boldsymbol{x}}_k-\lambda F^{\mathrm{VR}}(\boldsymbol{x}^k))$ in the first inequality and used the firmly nonexpansiveness of the projection and (19). It follows that

$$\|\bar{\boldsymbol{x}}_k-\boldsymbol{x}^k\|^2\ge\tfrac14\operatorname{res}(\boldsymbol{x}^k)^2-\tfrac12\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2-\lambda^2\|\varepsilon_k\|^2.$$

By substituting in (27), we have that

$$\begin{aligned}&\tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^*\|^2+\tfrac{1}{\delta}\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2\le\tfrac{1}{1-\delta}\|\boldsymbol{x}^*-\bar{\boldsymbol{x}}^k\|^2\\&\quad-\tfrac{1}{\delta}\left(\tfrac14\operatorname{res}(\boldsymbol{x}^k)^2-\tfrac12\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2-\lambda^2\|\varepsilon_k\|^2\right)\\&\quad+\ell\lambda\left(\|\boldsymbol{x}^k-\boldsymbol{x}^{k-1}\|^2+\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2\right)\\&\quad+\lambda\|\varepsilon^k-\varepsilon^{k-1}\|^2+\lambda\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2+2\lambda\langle\varepsilon^k,\boldsymbol{x}^*-\boldsymbol{x}^k\rangle.\end{aligned}$$

Finally, by taking the expected value, grouping and using Remark 2 and Assumptions 3 and 6, we have

$$\begin{aligned}&\mathbb{E}\left[\tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^*\|^2|\mathcal{F}_k\right]\\&\quad+\mathbb{E}\left[\left(\tfrac{1}{2\delta}-\ell\lambda-\lambda\right)\|\boldsymbol{x}^k-\boldsymbol{x}^{k+1}\|^2|\mathcal{F}_k\right]\\&\quad\le\tfrac{1}{1-\delta}\|\boldsymbol{x}^*-\bar{\boldsymbol{x}}^k\|^2+\ell\lambda\|\boldsymbol{x}^k-\boldsymbol{x}^{k-1}\|^2\\&\quad+\frac{2\lambda C\sigma}{N_k}+\frac{2\lambda C\sigma}{N_{k-1}}+\frac{\lambda}{\delta}\frac{C\sigma}{N_k}\\&\quad-\tfrac{1}{\delta}\|\boldsymbol{x}^k-\bar{\boldsymbol{x}}^k\|^2-\tfrac{1}{4\delta}\operatorname{res}(\boldsymbol{x}^k)^2.\end{aligned}$$

To use Lemma 1, let

$$\begin{aligned}\alpha_k&=\tfrac{1}{1-\delta}\|\boldsymbol{x}^*-\bar{\boldsymbol{x}}^k\|^2+\ell\lambda\|\boldsymbol{x}^k-\boldsymbol{x}^{k-1}\|^2\\\theta_k&=\tfrac{1}{\delta}\|\boldsymbol{x}^k-\bar{\boldsymbol{x}}^k\|^2+\tfrac{1}{4\delta}\operatorname{res}(\boldsymbol{x}^k)^2\\\eta_k&=\frac{2\lambda C\sigma}{N_k}+\frac{2\lambda C\sigma}{N_{k-1}}+\frac{\lambda}{\delta}\frac{C\sigma}{N_k}.\end{aligned}$$

Applying the Robbins Siegmund Lemma we conclude that α_k converges and that $\sum_{k\in\mathbb{N}}\theta_k$ is summable. This implies that the sequence $(\bar{\boldsymbol{x}}^k)_{k\in\mathbb{N}}$ is bounded and that $\|\boldsymbol{x}^k-\bar{\boldsymbol{x}}^k\|\to0$ (otherwise $\sum(1/\delta)\|\boldsymbol{x}^k-\bar{\boldsymbol{x}}^k\|^2=\infty$). Therefore $(\boldsymbol{x}^k)_{k\in\mathbb{N}}$ has at least one cluster point $\tilde{\boldsymbol{x}}$. Moreover, since $\sum_{k\in\mathbb{N}}\theta_k<\infty$, $\operatorname{res}(\boldsymbol{x}^k)^2\to0$ and $\operatorname{res}(\tilde{\boldsymbol{x}}^k)^2=0$.

Appendix C
Proof of Theorem 2

Proof of Theorem 2: We start by using the fact that the projection is firmly quasinonexpansive

$$\begin{aligned}&\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^*\|^2\\&\le\|\boldsymbol{x}^*-\tilde{\boldsymbol{x}}^k+\lambda F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)\|^2\\&\quad-\|\bar{\boldsymbol{x}}^k-\lambda F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)-\boldsymbol{x}^{k+1}\|^2\\&\le\|\boldsymbol{x}-\bar{\boldsymbol{x}}^k\|^2-\|\bar{\boldsymbol{x}}^k-\boldsymbol{x}^{k+1}\|^2+2\lambda_k\langle F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\boldsymbol{x}^*-\bar{\boldsymbol{x}}^k\rangle\\&\quad+2\lambda_k\langle F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\bar{\boldsymbol{x}}^k-\boldsymbol{x}^{k+1}\rangle\\&=\|\boldsymbol{x}^*-\bar{\boldsymbol{x}}^k\|^2-\|\bar{\boldsymbol{x}}^k-\boldsymbol{x}^{k+1}\|^2+2\lambda_k\langle F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\bar{\boldsymbol{x}}^k-\boldsymbol{x}^{k+1}\rangle\\&\quad+2\lambda_k\langle F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\boldsymbol{x}^*-\boldsymbol{x}^k\rangle+2\lambda_k\langle F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\boldsymbol{x}^k-\bar{\boldsymbol{x}}^k\rangle.\end{aligned}$$

Now we apply Lemma 2.2 and Lemma 2.3 to $\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^*\|$

$$\begin{aligned}\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^*\|^2=\tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^*\|^2-\tfrac{\delta}{1-\delta}\|\bar{\boldsymbol{x}}^k-\boldsymbol{x}^*\|^2\\+\delta\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\|^2.\end{aligned}\tag{28}$$

Then, we can rewrite the inequality as

$$\begin{aligned}&\tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^*\|^2\le\tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^k-\boldsymbol{x}^*\|^2\\&\quad+2\lambda_k\langle F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\boldsymbol{x}^*-\boldsymbol{x}^k\rangle+2\lambda_k\langle F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\boldsymbol{x}^k-\bar{\boldsymbol{x}}^k\rangle\\&\quad+2\lambda_k\langle F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\bar{\boldsymbol{x}}^k-\boldsymbol{x}^{k+1}\rangle-(\delta+1)\|\boldsymbol{x}^{k+1}-\bar{\boldsymbol{x}}^k\|^2.\end{aligned}\tag{29}$$

By applying Young's inequality we obtain

$$\begin{aligned}&2\lambda_k\langle F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\boldsymbol{x}^k-\bar{\boldsymbol{x}}^k\rangle\\&\quad\le\lambda_k^2\|F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)\|^2+\|\boldsymbol{x}^k-\bar{\boldsymbol{x}}^k\|^2\\&2\lambda_k\langle F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\bar{\boldsymbol{x}}^k-\boldsymbol{x}^{k+1}\rangle\\&\quad\le\lambda_k^2\|F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)\|^2+\|\bar{\boldsymbol{x}}^k-\boldsymbol{x}^{k+1}\|^2.\end{aligned}$$

Then, inequality (29) becomes

$$\begin{aligned}&\tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^*\|^2\le\tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^k-\boldsymbol{x}^*\|^2\\&\quad+2\lambda_k\langle F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\boldsymbol{x}^*-\boldsymbol{x}^k\rangle\\&\quad+2\lambda_k^2\|F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)\|^2-(\delta+1)\|\boldsymbol{x}^{k+1}-\bar{\boldsymbol{x}}^k\|^2\\&\quad+\|\boldsymbol{x}^k-\bar{\boldsymbol{x}}^k\|^2+\|\bar{\boldsymbol{x}}^k-\boldsymbol{x}^{k+1}\|^2.\end{aligned}\tag{30}$$

Reordering, adding and subtracting $2\lambda_k\langle\mathbb{F}(\boldsymbol{x}^k),\boldsymbol{x}^k-\boldsymbol{x}^*\rangle$ and using Lemma 2, we obtain

$$\begin{aligned}&\tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^*\|^2+\delta\|\boldsymbol{x}^{k+1}-\bar{\boldsymbol{x}}^k\|^2\le\tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^k-\boldsymbol{x}^*\|^2\\&\quad+2\lambda_k\langle\mathbb{F}(\boldsymbol{x}^k)-F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k),\boldsymbol{x}^k-\boldsymbol{x}^*\rangle-2\lambda_k\langle\mathbb{F}(\boldsymbol{x}^k),\boldsymbol{x}^k-\boldsymbol{x}^*\rangle\\&\quad+2\lambda_k^2\|F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)\|^2+\delta^2\|\boldsymbol{x}^k-\boldsymbol{x}^{k-1}\|^2.\end{aligned}\tag{31}$$

Then, by the definition of ϵ^k, reordering leads to

$$\begin{aligned}2\lambda_k&\langle\mathbb{F}(\boldsymbol{x}^k), \boldsymbol{x}^k-\boldsymbol{x}^*\rangle\\ &\leq \tfrac{1}{1-\delta}(\|\bar{\boldsymbol{x}}^k-\boldsymbol{x}^*\|^2-\|\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^*\|^2)\\ &\quad+\delta(\|\boldsymbol{x}^k-\boldsymbol{x}^{k-1}\|^2-\|\boldsymbol{x}^{k+1}-\bar{\boldsymbol{x}}^k\|^2)\\ &\quad+2\lambda_k^2\|F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)\|^2+2\lambda_k\langle\epsilon^k,\boldsymbol{x}^k-\boldsymbol{x}^*\rangle. \end{aligned}\tag{32}$$

Next, we sum over all the iterations, hence inequality (32) becomes

$$\begin{aligned}2\sum_{k=1}^K\lambda_k\langle\mathbb{F}(\boldsymbol{x}^k), \boldsymbol{x}^k-\boldsymbol{x}^*\rangle &\leq 2\sum_{k=1}^K\lambda_k\langle\epsilon^k,\boldsymbol{x}^k-\boldsymbol{x}^*\rangle\\ &\leq \tfrac{1}{1-\delta}\sum_{k=1}^K(\|\bar{\boldsymbol{x}}^k-\boldsymbol{x}^*\|^2-\|\bar{\boldsymbol{x}}^{k+1}-\boldsymbol{x}^*\|^2)\\ &\quad+\delta\sum_{k=1}^K\big(\|\boldsymbol{x}^k-\boldsymbol{x}^{k-1}\|^2-\|\boldsymbol{x}^{k+1}-\bar{\boldsymbol{x}}^k\|^2\big)\\ &\quad+2\sum_{k=1}^K\lambda_k^2\|F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)\|^2.\end{aligned}\tag{33}$$

Using Assumption 2 and resolving the sums, we obtain

$$\begin{aligned}2\sum_{k=1}^K\lambda_k\langle\mathbb{F}(\boldsymbol{x}^*), \boldsymbol{x}^k-\boldsymbol{x}^*\rangle &\leq 2\sum_{k=1}^K\lambda_k\langle\epsilon^k,\boldsymbol{x}^k-\boldsymbol{x}^*\rangle\\ &\leq \tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^0-\boldsymbol{x}^*\|^2+\delta\|\boldsymbol{x}^0-\bar{\boldsymbol{x}}^{-1}\|^2\\ &\quad+2\sum_{k=1}^K\lambda_k^2\|F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)\|^2.\end{aligned}\tag{34}$$

Now, we note that $\langle\epsilon^k,\boldsymbol{x}^k-\boldsymbol{x}^*\rangle=\langle\epsilon^k,\boldsymbol{x}^k-\boldsymbol{u}^k\rangle+\langle\epsilon^k,\boldsymbol{u}^k-\boldsymbol{x}^*\rangle$. We define $\boldsymbol{u}^0=\boldsymbol{x}^0$ and $\boldsymbol{u}^{k+1}=\mathrm{proj}(\boldsymbol{u}^k-\lambda_k\epsilon^k)$, thus

$$\begin{aligned}\|\boldsymbol{u}^{k+1}-\boldsymbol{x}^*\|^2&=\|\mathrm{proj}(\boldsymbol{u}^k-\lambda_k\epsilon^k)-\boldsymbol{x}^*\|^2\\ &\leq\|\boldsymbol{u}^k-\lambda_k\epsilon^k-\boldsymbol{x}^*\|^2\\ &\leq\|\boldsymbol{u}^k-\boldsymbol{x}^*\|^2+\lambda_k\|\epsilon^k\|^2-2\lambda_k\langle\epsilon^k,\boldsymbol{u}^k-\boldsymbol{x}^*\rangle.\end{aligned}\tag{35}$$

Therefore, $2\lambda_k\langle\epsilon^k,\boldsymbol{x}^k-\boldsymbol{x}^*\rangle=2\lambda_k\langle\epsilon^k,\boldsymbol{x}^k-\boldsymbol{u}^k\rangle+\|\boldsymbol{u}^k-\boldsymbol{x}^*\|^2+\lambda_k\|\epsilon^k\|^2-\|\boldsymbol{u}^{k+1}-\boldsymbol{x}^*\|^2$. By including this in (34) and by doing the sum, we obtain

$$\begin{aligned}2\sum_{k=1}^K&\lambda_k\langle\mathbb{F}(\boldsymbol{x}^*), \boldsymbol{x}^k-\boldsymbol{x}^*\rangle\\ &\leq \tfrac{1}{1-\delta}\|\bar{\boldsymbol{x}}^0-\boldsymbol{x}^*\|^2+\delta\|\boldsymbol{x}^0-\bar{\boldsymbol{x}}^{-1}\|^2\\ &\quad+2\sum_{k=1}^K\lambda_k^2\|F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)\|^2+\sum_{k=1}^K\lambda_k^2\|\epsilon^k\|^2\\ &\quad+\|\boldsymbol{u}_0-\boldsymbol{x}^*\|^2+2\sum_{k=1}^K\lambda_k\langle\epsilon^k,\boldsymbol{x}^k-\boldsymbol{u}^k\rangle.\end{aligned}\tag{36}$$

By definition, $\|\boldsymbol{u}^0-\boldsymbol{x}^*\|^2=\|\boldsymbol{x}^0-\boldsymbol{x}^*\|^2$. Then, by taking the expected value in (36) and using Assumption 3, we conclude that

$$\begin{aligned}2\sum_{k=1}^K&\lambda_k\langle\mathbb{F}(\boldsymbol{x}^*), \boldsymbol{x}^k-\boldsymbol{x}^*\rangle\\ &\leq (\tfrac{1}{1-\delta}+1)\|\bar{\boldsymbol{x}}^0-\boldsymbol{x}^*\|^2+\delta\|\boldsymbol{x}^0-\bar{\boldsymbol{x}}^{-1}\|^2\\ &\quad+2\sum_{k=1}^K\lambda_k^2\mathbb{E}\big[\|F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)\|^2|\mathcal{F}_k\big]\\ &\quad+\sum_{k=1}^K\lambda_k^2\mathbb{E}\big[\|\epsilon^k\|^2|\mathcal{F}_k\big].\end{aligned}\tag{37}$$

Let us define $S=\sum_{k=1}^K\lambda_k$, $\boldsymbol{X}^K=\big(\sum_{k=1}^K\lambda_k\boldsymbol{x}^k\big)/\big(\sum_{k=1}^K\lambda_k\big)=(1/S)\sum_{k=1}^K\lambda_k\boldsymbol{x}^k$. Then,

$$\begin{aligned}2S\langle\mathbb{F}(\boldsymbol{x}^*),\boldsymbol{X}^K-\boldsymbol{x}^*\rangle &\leq \tfrac{2-\delta}{1-\delta}\|\bar{\boldsymbol{x}}^0-\boldsymbol{x}^*\|^2+\delta\|\boldsymbol{x}^0-\bar{\boldsymbol{x}}^{-1}\|^2\\ &\quad+2\sum_{k=1}^K\lambda_k^2\mathbb{E}\big[\|F^{\mathrm{SA}}(\boldsymbol{x}^k,\xi^k)\|^2|\mathcal{F}_k\big]\\ &\quad+\sum_{k=1}^K\lambda_k^2\mathbb{E}\big[\|\epsilon^k\|^2|\mathcal{F}_k\big]\\ &\leq\tfrac{2-\delta^2}{1-\delta}R+\big(2B^2+\sigma^2\big)\sum_{k=1}^K\lambda_k.\end{aligned}\tag{38}$$

Finally, it holds that if λ_k is constant, $S=K\lambda$ and $\sum_{k=1}^K\lambda_k^2=K\lambda^2$

$$\langle\mathbb{F}(\boldsymbol{x}^*),\boldsymbol{X}^K-\boldsymbol{x}^*\rangle\leq\frac{cR}{K\lambda}+\big(2B^2+\sigma^2\big)\lambda.$$

□

References

[1] I. Goodfellow, “NIPS 2016 tutorial: Generative adversarial networks,” 2017, *arXiv:1701.00160*. [Online]. Available: http://arxiv.org/abs/1701.00160

[2] I. Goodfellow *et al.*, “Generative adversarial nets,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[3] Z. Wang, Q. She, and T. E. Ward, “Generative adversarial networks in computer vision: A survey and taxonomy,” 2019, *arXiv:1906.01529*. [Online]. Available: http://arxiv.org/abs/1906.01529

[4] Q. Kang, S. Yao, M. Zhou, K. Zhang, and A. Abusorrah, “Effective visual domain adaptation via generative adversarial distribution matching,” *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 10, 2020, doi: 10.1109/TNNLS.2020.3016180.

[5] X. Song, Y. Chen, Z.-H. Feng, G. Hu, D.-J. Yu, and X.-J. Wu, “SP-GAN: Self-growing and pruning generative adversarial networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2458–2469, Jun. 2021.

[6] Q. Yang *et al.*, “Low-dose CT image denoising using a generative adversarial network with Wasserstein distance and perceptual loss,” *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1348–1357, Jun. 2018.

[7] N. Eldeen M. Khalifa, M. Hamed N. Taha, A. E. Hassanien, and S. Elghamrawy, “Detection of coronavirus (COVID-19) associated pneumonia based on generative adversarial networks and a fine-tuned deep transfer learning model using chest X-ray dataset,” 2020, *arXiv:2004.01184*. [Online]. Available: http://arxiv.org/abs/2004.01184

[8] H. Zhang, V. Sindagi, and V. M. Patel, “Image de-raining using a conditional generative adversarial network,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 11, pp. 3943–3956, Nov. 2020.

[9] P. Xiang, L. Wang, F. Wu, J. Cheng, and M. Zhou, “Single-image de-raining with feature-supervised generative adversarial network,” *IEEE Signal Process. Lett.*, vol. 26, no. 5, pp. 650–654, May 2019.

[10] K. Liu, Z. Ye, H. Guo, D. Cao, L. Chen, and F.-Y. Wang, “FISS GAN: A generative adversarial network for foggy image semantic segmentation,” *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 8, pp. 1428–1439, Aug. 2021.

[11] K. Zhang, Y. Su, X. Guo, L. Qi, and Z. Zhao, “MU-GAN: Facial attribute editing based on multi-attention mechanism,” *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 9, pp. 1614–1626, Sep. 2021.

[12] Y. Chen *et al.*, “Research on image inpainting algorithm of improved GAN based on two-discriminations networks,” *Appl. Intell.*, vol. 51, no. 6, pp. 3460–3474, 2021.

[13] Y. Chen *et al.*, “The improved image inpainting algorithm via encoder and similarity constraint,” *Vis. Comput.*, vol. 37, pp. 1691–1705, Jul. 2021.

[14] S. W. Kim, Y. Zhou, J. Philion, A. Torralba, and S. Fidler, “Learning to simulate dynamic environments with GameGAN,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 1231–1240.

[15] S. R. Bulò, B. Biggio, I. Pillai, M. Pelillo, and F. Roli, "Randomized prediction games for adversarial machine learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2466–2478, Nov. 2017.
[16] F. A. Oliehoek, R. Savani, J. Gallego-Posada, E. van der Pol, E. D. de Jong, and R. Gross, "GANGs: Generative adversarial network games," 2017, *arXiv:1712.00679*. [Online]. Available: http://arxiv.org/abs/1712.00679
[17] M. Heusel *et al.*, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6626–6637.
[18] E. Mazumdar, L. J. Ratliff, and S. S. Sastry, "On gradient-based learning in continuous games," *SIAM J. Math. Data Sci.*, vol. 2, no. 1, pp. 103–131, Jan. 2020.
[19] G. Gidel, H. Berard, G. Vignoud, P. Vincent, and S. Lacoste-Julien, "A variational inequality perspective on generative adversarial networks," 2018, *arXiv:1802.10551*. [Online]. Available: http://arxiv.org/abs/1802.10551
[20] F. Facchinei and J.-S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems*. New York, NY, USA: Springer, 2007.
[21] Q. Tao, Q.-K. Gao, D.-J. Chu, and G.-W. Wu, "Stochastic learning via optimizing the variational inequalities," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 10, pp. 1769–1778, Oct. 2014.
[22] P. Mertikopoulos, B. Lecouat, H. Zenati, C.-S. Foo, V. Chandrasekhar, and G. Piliouras, "Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile," 2018, *arXiv:1807.02629*. [Online]. Available: http://arxiv.org/abs/1807.02629
[23] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 1951.
[24] R. E. Bruck, Jr., "On the weak convergence of an ergodic iteration for the solution of variational inequalities for monotone operators in Hilbert space," *J. Math. Anal. Appl.*, vol. 61, no. 1, pp. 159–164, Nov. 1977.
[25] A. N. Iusem, A. Jofré, R. I. Oliveira, and P. Thompson, "Extragradient method with variance reduction for stochastic variational inequalities," *SIAM J. Optim.*, vol. 27, no. 2, pp. 686–724, Jan. 2017.
[26] F. Yousefian, A. Nedić, and U. V. Shanbhag, "Optimal robust smoothing extragradient algorithms for stochastic variational inequality problems," in *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 5831–5836.
[27] K. Mishchenko, D. Kovalev, E. Shulgin, P. Richtarik, and Y. Malitsky, "Revisiting stochastic extragradient," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 4573–4582.
[28] R. I. Boţ, P. Mertikopoulos, M. Staudigl, and P. T. Vuong, "Minibatch forward-backward-forward methods for solving stochastic variational inequalities," *Stochastic Syst.*, vol. 11, no. 2, pp. 112–139, Jun. 2021.
[29] F. Facchinei and C. Kanzow, "Generalized Nash equilibrium problems," *Ann. Oper. Res.*, vol. 175, no. 1, pp. 177–211, 2010.
[30] Y. Malitsky, "Golden ratio algorithms for variational inequalities," *Math. Program.*, vol. 184, nos. 1–2, pp. 383–410, Nov. 2020, doi: 10.1007/s10107-019-01416-w.
[31] B. Franci and S. Grammatico, "Stochastic generalized Nash equilibrium seeking in merely monotone games," 2020, *arXiv:2002.08318*. [Online]. Available: http://arxiv.org/abs/2002.08318
[32] B. Franci and S. Grammatico, "A game–theoretic approach for generative adversarial networks," in *Proc. 59th IEEE Conf. Decis. Control (CDC)*, Dec. 2020, pp. 1646–1651.
[33] B. Franci and S. Grammatico, "A distributed forward-backward algorithm for stochastic generalized Nash equilibrium seeking," *IEEE Trans. Autom. Control*, early access, Dec. 25, 2020, doi: 10.1109/TAC.2020.3047369.
[34] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for GANs do actually converge?" 2018, *arXiv:1801.04406*. [Online]. Available: http://arxiv.org/abs/1801.04406
[35] S. Grammatico, "Comments on 'distributed robust adaptive equilibrium computation for generalized convex games' [automatica 63 (2016) 82–91]," *Automatica*, vol. 97, pp. 186–188, Nov. 2018.
[36] P. Tseng, "A modified forward-backward splitting method for maximal monotone mappings," *SIAM J. Control Optim.*, vol. 38, no. 2, pp. 431–446, 2000.
[37] R. I. Bot, M. Sedlmayer, and P. T. Vuong, "A relaxed inertial forward-backward-forward algorithm for solving monotone inclusions with application to GANs," 2020, *arXiv:2003.07886*. [Online]. Available: http://arxiv.org/abs/2003.07886
[38] G. Korpelevich, "The extragradient method for finding saddle points and other problems," *Matecon*, vol. 12, no. 4, pp. 747–756, 1976.
[39] Y. Yazıcı, C.-S. Foo, S. Winkler, K.-H. Yap, G. Piliouras, and V. Chandrasekhar, "The unusual effectiveness of averaging in GAN training," 2018, *arXiv:1806.04498*. [Online]. Available: http://arxiv.org/abs/1806.04498
[40] L. Mescheder, S. Nowozin, and A. Geiger, "The numerics of GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1825–1835.
[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[42] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017, *arXiv:1701.07875*. [Online]. Available: http://arxiv.org/abs/1701.07875
[43] F. Facchinei, A. Fischer, and V. Piccialli, "On generalized Nash games and variational inequalities," *Oper. Res. Lett.*, vol. 35, no. 2, pp. 159–164, 2007.
[44] U. Ravat and U. V. Shanbhag, "On the characterization of solution sets of smooth and nonsmooth convex stochastic Nash games," *SIAM J. Optim.*, vol. 21, no. 3, pp. 1168–1199, Jul. 2011.
[45] J. Koshal, A. Nedic, and U. V. Shanbhag, "Regularized iterative stochastic approximation methods for stochastic variational inequality problems," *IEEE Trans. Autom. Control*, vol. 58, no. 3, pp. 594–609, Mar. 2013.
[46] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*. [Online]. Available: http://arxiv.org/abs/1511.06434
[47] 'A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," M.S. thesis, Univ. Toronto, Toronto, ON, Canada, 2009.
[48] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2234–2242.
[49] H. H. Bauschke *et al.*, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, vol. 408. New York, NY, USA: Springer, 2011.
[50] H. Robbins and D. Siegmund, "A convergence theorem for non negative almost supermartingales and some applications," in *Optimizing Methods in Statistics*. Amsterdam, The Netherlands: Elsevier, 1971, pp. 233–257.

Barbara Franci received the bachelor's and master's degrees in mathematics from the University of Siena, Siena, Italy, in 2012 and 2014, respectively, and the Ph.D. degree from the Politecnico of Turin and University of Turin, Turin, Italy, in 2018.

From September to December 2016, she visited the Department of Mechanical Engineering, University of California, Santa Barbara, CA, USA. She is currently a Post-Doctoral Researcher with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. Her current research interests are on game theory and its applications.

Dr. Franci was awarded the Ph.D. Quality Award by the Academic Board of Politecnico di Torino, in 2017.

Sergio Grammatico (Senior Member, IEEE) was born in Italy in 1987. He received the bachelor's degree *(summa cum laude)* in computer engineering, the master's degree *(summa cum laude)* in automatic control engineering, and the Ph.D. degree in automatic control from the University of Pisa, Pisa, Italy, in February 2008, October 2009, and March 2013, respectively, and the master's degree *(summa cum laude)* in engineering science from the Sant'Anna School of Advanced Studies, Pisa, Italy, in November 2011.

From February to April 2010 and November to December 2011, he visited the Department of Mathematics, University of Hawaii at Manoa, USA. From January to July 2012, he visited the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA. From 2013 to 2015, he was a Post-Doctoral Research Fellow with the Automatic Control Laboratory, ETH Zurich, Zurich, Switzerland. From 2015 to 2018, he was an Assistant Professor with the Department of Electrical Engineering, Control Systems, TU Eindhoven, The Netherlands, and with the Delft Center for Systems and Control, TU Delft, Delft, The Netherlands. He is currently an Associate Professor with the Delft Center for Systems and Control, Delft University of Technology, Delft.