A Convergent and Dimension-Independent Min-Max Optimization Algorithm

Vijay Keswani
Yale University

Oren Mangoubi
WPI

Sushant Sachdeva
University of Toronto

Nisheeth K. Vishnoi
Yale University

July 4, 2022

Abstract

We study a variant of a recently introduced min-max optimization framework where the max-player is constrained to update its parameters in a greedy manner until it reaches a first-order stationary point. Our equilibrium definition for this framework depends on a proposal distribution which the min-player uses to choose directions in which to update its parameters. We show that, given a smooth and bounded nonconvex-nonconcave objective function, access to any proposal distribution for the min-player's updates, and stochastic gradient oracle for the max-player, our algorithm converges to the aforementioned approximate local equilibrium in a number of iterations that does not depend on the dimension. The equilibrium point found by our algorithm depends on the proposal distribution, and when applying our algorithm to train GANs we choose the proposal distribution to be a distribution of stochastic gradients. We empirically evaluate our algorithm on challenging nonconvex-nonconcave test-functions and loss functions arising in GAN training. Our algorithm converges on these test functions and, when used to train GANs, trains stably on synthetic and real-world datasets and avoids mode collapse.

Contents

1 Introduction

For a loss function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ on some (convex) domain $\mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^d \times \mathbb{R}^d$, we consider:

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y). \tag{1}$$

This min-max optimization problem has several applications to machine learning, including GANs [22] and adversarial training [36]. In many of these applications, only first-order access to f is available efficiently, and gradient-based algorithms are widely used.

Unfortunately, there is a lack of gradient-based algorithms with convergence guarantees for this min-max framework if one allows for loss functions $f(x, y)$ which are nonconvex and nonconcave in x and y respectively. This lack of convergence guarantees can be a serious problem in practice, since popular algorithms such as gradient descent-ascent (GDA) oftentimes fail to converge, and GANs trained with these algorithms can suffer from issues such as cycling [2] and "mode collapse" [16, 8, 51].

Since min-max optimization includes minimization (and maximization) as special cases, it is intractable for general nonconvex-nonconcave functions. Motivated by the success of a long line of results which show efficient convergence of minimization algorithms to various (approximate) local minimum notions (e.g., [46, 20, 1]), previous works have sought to extend these ideas of local minima to various (approximate) notions of local min-max point–that is, a point $(x^\star, y^\star)$ where $x^\star$ is a local minimum of $f(\cdot, y^\star)$ and $y^\star$ is a local maximum of $f(x^\star, \cdot)$– in the hope that this will allow for algorithms with convergence guarantees to such points. Unfortunately, to prove convergence, these works (e.g., Nemirovski [44], Lu et al. [35]) make strong assumptions on f, e.g. assume $f(x, y)$ is concave in y, or that their algorithm is given a starting point such that its underlying dynamical system converges (Heusel et al. [23], Mescheder et al. [39]). It is a challenge to develop gradient-based algorithms which converge efficiently to an equilibrium for even a local variant of the min-max framework under less restrictive assumptions comparable to those required for convergence of algorithms to local minima.

Our contributions. We study a variant of the min-max framework which allows the max-player to update y in a "greedy" manner (Section 3.1). This greedy restriction models first-order maximization algorithms such as gradient ascent, popular in machine learning applications, which can make updates far from the current value of y when run for multiple steps. Roughly, from the current point (x, y), our framework allows the max-player to update y along *any* continuous path, along which the loss $f(x, \cdot)$ is nondecreasing.

Our main contribution is a new gradient-based algorithm (Algorithm 1) that provably converges from any initial point to an approximate local equilibrium for this framework (Definition 3.2). Our approximate equilibrium definition depends on the choice of proposal distribution $Q_{x,y}$, parametrized by (x, y), which the min-player uses to update its parameters at any given point (x, y). In particular, for a b-bounded function f with L-Lipschitz gradient, and $\varepsilon \geq 0$, our algorithm requires $\text{poly}(b, L, 1/\varepsilon)$ gradient and function oracle calls to converge to an (ε, Q)-approximate local equilibrium (Theorem 3.3). The number of oracle calls required by our algorithm is independent of the dimension d. Gradient-based algorithms that converge in (almost) dimension independent number of iterations are required for machine-learning applications where the dimension d, equal to the number of trainable parameters, can be very large.

The equilibrium point our algorithm converges to depends on the choice of proposal distribution Q. In the special case when $f(x, y)$ is strongly convex in x and strongly concave in y, there is a

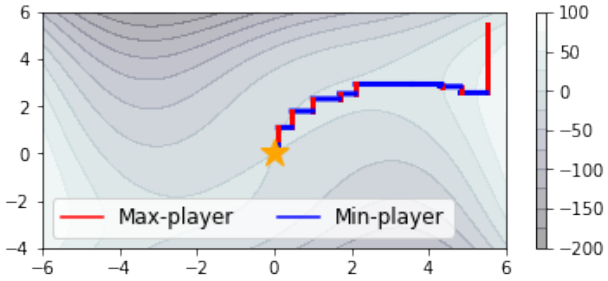

Figure 1: Our algorithm applied to the function $f(x, y) = (4x^2 - (y - 3x + 0.05x^3)^2 - 0.1y^4)e^{-0.01(x^2+y^2)}$ with global min-max point $(x, y) = (0, 0)$ (yellow star). Our algorithm's max-player (red segments) first finds a point where $f(x, \cdot)$ is maximized. The min-player then proposes random updates to x, and only accepts those updates which lead to a decrease in the value of $f(x, y)$ after the max-player's response is taken into account (blue segments). This allows our algorithm to converge to $(0, 0)$. This function is considered as a challenging test function in [56], who show several first-order algorithms, namely GDA, OMD, and extra-gradient method [28], fail to converge on this function and instead cycle forever (see Figure 2).

choice of proposal distribution Q– the (deterministic or stochastic) gradients with mean $-\nabla_x f$–for which our (ε, Q)-equilibrium corresponds to an (approximate) global min-max point with duality gap roughly $O(\varepsilon)$ (Theorem A.2 in the appendix). Our algorithm can find such a point in time that is polynomial in $\frac{1}{\varepsilon}$ and independent of dimension (Corollary A.10). This motivates using (stochastic) gradients for proposal distributions in more general settings as well, and, when training GANs we choose the proposal distribution to be the distribution of stochastic gradients.

Empirically, we show that our algorithm converges on test functions [56] on which other popular gradient-based min-max optimization algorithms such as GDA and optimistic mirror descent (OMD) [13] are known to either diverge or cycle (Figure 1, see also Figure 2 in Section 5.1). We also show that a practical variant of our algorithm can be employed for training GANs, with a per-step complexity and memory requirement similar to GDA. We observe that our algorithm consistently learns a greater number of modes than GDA, OMD, and unrolled GANs (Table 1), when applied to training GANs on a Gaussian mixture dataset. While not the focus of this paper, we also provide results for our algorithm on the real-world datasets in the Supplementary Material.

Discussion of equilibrium. The equilibrium points $(x^\star, y^\star)$ our algorithm converges to can be viewed as local equilibria for a game where the maximizing player is restricted to making greedy updates to the value of y. Namely, the point $x^\star$ is an approximate local minimum of an alternative to the function $\max_y f(\cdot, y)$ where, rather than maximizing over all $y \in \mathbb{R}^d$, the maximum is instead taken over all "greedy" paths–i.e., paths along which $f(x^\star, \cdot)$ is increasing–initialized at the value $y^\star$. Additionally, $y^\star$ is an approximate local maximum of $f(x^\star, \cdot)$. In particular, we show that any point $(x^\star, y^\star)$ which is a local min-max point is also an equilibrium point for our algorithm (see Section 3.1).

Discussion of assumptions. For our main result, we assume f is bounded above and below. The assumption that the loss function is bounded below is standard in the minimization literature (see e.g., [46]), as an unbounded function need not achieve its minimum and a minimization algorithm could diverge in a manner such that the loss function value tends to $-\infty$. Thus, in min-max

optimization, both the upper and lower bound assumptions are necessary to ensure the existence of even an (approximate) global min-max point. If we drop the lower bound assumption, and only assume $f(x, \cdot)$ is bounded above for $x \in \mathbb{R}^d$, our algorithm still does not cycle: instead it either converges to a local equilibrium $(x^\star, y^\star)$, or the value of f diverges monotonically to $-\infty$. Such functions include popular losses, e.g. cross entropy [22] which is bounded above by zero, making our algorithm applicable to training GANs.

2 Related Work

Local frameworks. In addition to the local min-max framework (e.g., [44, 35, 23, 39]), previous works propose local frameworks where the max-player is able to choose its move after the min-player. These include the local stackleberg equilibrium [18] and the closely related local minimax point [24]. In the local min-max, local stackleberg and local minimax frameworks, the max-player is restricted to move in a small ball around the current point y. In contrast, in our framework, the max-player can move much farther, as long as it follows a path along which f is continuously increasing.

Convergence guarantees. Several works have studied the convergence properties of GDA dynamics [43, 39, 4, 11, 24, 30], and established that GDA suffers from severe limitations: GDA can exhibit rotation around some points, or otherwise fail to converge. To address convergence issues for GDA, multiple works analyze algorithms based on Optimistic Mirror Descent (OMD), Extra-gradient (EG) methods, or similar approaches [21, 11, 31, 12, 42]. For instance, [13] guarantee convergence of OMD to a global min-max point on bilinear losses $f(x, y) = x^\top A y$, and [41] also show convergence of OMD and EG methods on strongly convex-strongly concave f. However, as observed in [56], GDA, OMD, and EG fail to converge on some simple nonconvex-nonconcave test functions; in comparison, we observe that our algorithm converges for these functions (Figure 2). Many works make additional assumptions to prove convergence–[38] show asymptotic convergence of OMD under a "coherence" assumption, and [4] show convergence of a second-order algorithm if f corresponds to a Hamiltonian game. Some works assume there exists a "variational inequality" solution $(x^\star, y^\star)$ such that, roughly, the component of the gradient field $(-\nabla_x f, \nabla_y f)$ in the direction away from $(x^\star, y^\star)$ is very small [9, 33, 52, 14, 34]. Other works [57] assume a "PL" condition which says that magnitude of $\nabla_x f(x, y)$ is at least $f(x, y) - \min_x f(x, y)$. However, many simple functions, e.g. $f(x, y) = \sin(x)\sin(y)$, do not satisfy Hamiltonian game, coherence, variational, and PL assumptions.

For this reason, multiple works show convergence to an (approximate) local min-max point. For instance [23] prove convergence of finite-step GDA to a local min-max point, under the assumption that their algorithm is initialized such that the underlying continuous dynamics converge to a local min-max point. And [39] show convergence if their algorithm is initialized in a small neighborhood of a local min-max point. In addition, many works provide convergence guarantees to a local *stackleberg* or local *minimax* point, if their algorithm is provided with a starting point in the region of attraction [17, 18], or a small enough neighborhood [56], of such an equilibrium. And other works [45, 26, 44, 49, 35, 32, 47, 54, 27] show convergence to an approximate local min-max point when f may be nonconvex in x, but is concave in y. In contrast to the above works, our algorithm is guaranteed to converge for any nonconvex-nonconcave f, from any starting point, in a number of gradient evaluations that is independent of the dimension d and polynomial in L and b if f is b-bounded with L-Lipschitz gradient. Such smoothness/Lipschitz bounds are standard

in convergence guarantees for optimization algorithms [7, 20, 55]. Similar to our approach, some algorithms make multiple y-player updates each iteration (e.g., [47]). However, these algorithms are not guaranteed to converge from any initial point on any nonconvex-noncocave smooth bounded f; to overcome this, our algorithm introduces a randomized accept-reject procedure.

Greedy paths. [37] also consider a framework where the max-player makes updates in a greedy manner. The "greedy paths" considered in their work are defined such that at every point along these paths, f is non-decreasing, and the first derivative of f is at least ε or the 2nd derivative is at least $\sqrt{\varepsilon}$. In contrast, we just require a condition on the first derivative of f along the path. This distinction gives rise to a different framework and equilibrium than the one presented in their work. Secondly, [37] is a second-order method that converges to an ε-approximate local equilibrium in $\text{poly}(d, b, L, 1/\varepsilon)$ Hessian evaluations. On the other hand, the convergence of our algorithm is independent of d; it requires $\text{poly}(b, L, 1/\varepsilon)$ gradient evaluations for convergence.

Training GANs. An important line of work focuses on designing min-max optimization algorithms that mitigate non-convergence behavior such as cycling when training GANs using GDA [22]. [13] show OMD can mitigate cycling when training GANs with Wasserstein loss. In contrast to both GDA and OMD, where at each iteration the min- and max-players are allowed only to make small updates roughly proportional to their respective gradients, our algorithm empowers the max-player to make large updates at each iteration. [40] introduced Unrolled GANs, where the min-player optimizes an "unrolled" loss that allows the min-player to simulate a fixed number of max-player updates. While this has some similarity to our algorithm the main distinction is that the min-player in Unrolled GANs may not reach an (approximate) local minimum, and hence their algorithm does not have any convergence guarantees. We observe that our algorithm, applied to training GANs, trains stably and avoids mode collapse.

3 Theoretical Results

As a first step towards obtaining a computationally tractable variant of the min-max framework, we consider the local min-max point studied in prior work— that is, any point (x, y) such that x is local minimum of $f(\cdot, y)$ and y is a local maximum of $f(x, \cdot)$. Unfortunately, local min-max points may not exist even on smooth and bounded functions. For instance, the function $f(x, y) = \sin(x + y)$ has no local min-max points. This is because, while $f(x, y) = \sin(x + y)$ has local maximum in y at all points along the collection of lines $S = \{(x, y) : x + y = \frac{\pi}{2} + 2\pi k, k \in \mathbb{N}\}$, $\sin(x + y)$ does not have a local minimum in x at any of these points. However, the points S are all global min-max points of $f(x, y) = \sin(x + y)$, since for every $(x, y) \in S$, x is a global minimum of $\max_{y \in \mathbb{R}^d} f(\cdot, y)$, and y is a global maximum of $f(x, \cdot)$. This is true even though x is neither a global nor a local minimum of $f(\cdot, y)$.

On the other hand, an (approximate) *global* min-max point is always guaranteed to exist for smooth and bounded f. This is because, in the global min-max framework, before the min-player considers whether to choose a value x, it is able to "look ahead" and anticipate the response $\arg\max_{y \in \mathbb{R}^d} f(x, y)$ of the max-player. Thus, for any smooth and bounded function f, one can always find an (approximate) global min-max point by first finding an (approximate) global minimum x of the function $\max_{y \in \mathbb{R}^d} f(\cdot, y)$, and then finding a value of y which maximizes $f(x, \cdot)$. In order to guarantee existence for our framework, we would therefore ideally like to allow the min-player to

anticipate the max-player's response $\max_{y\in\mathbb{R}^d} f(\cdot, y)$ to any value of x proposed by the min-player. Unfortunately, computing the global maximum $\max_{y\in\mathbb{R}^d} f(\cdot, y)$ is intractable.

3.1 Framework and Equilibrium

To get around this problem, we consider a variant of the min-max framework, which empowers the max-player to update y in a "greedy" manner. More specifically, we restrict the max-player to update the current point (x, y) to any point in a set $P(x, y)$ consisting of the endpoints of paths in $\mathcal{Y}$ initialized at y along which $f(x, \cdot)$ is nondecreasing. These paths model the paths taken by a class of first-order algorithms, which includes popular algorithms such as gradient ascent. Our framework therefore allows the min-player to learn from max-players which are computationally tractable and yet (in contrast to the local min-max framework) are still empowered to make updates to the value of y which may lead to large increases in $f(x, y)$. Given a bounded loss $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, where $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}^d$ are convex, an equilibrium for our framework is a point $(x^\star, y^\star) \in \mathcal{X} \times \mathcal{Y}$ such that

$$x^\star \in \operatorname{argmin}_{x\in\mathcal{X}}(\max_{y\in P(x,y^*)} f(x, y)), \tag{2}$$

$$y^\star \in \operatorname{argmax}_{y\in P(x^\star,y^\star)} f(x^\star, y). \tag{3}$$

This is in contrast to the (global) min-max framework of (1) where the maximum is taken over all $y \in \mathcal{Y}$. However, solutions to (2) and (3) may not exist, and even when they do exist, finding such a solution is intractable since (2) generalizes nonconvex minimization.

Local equilibrium. Replacing the global minimum in (2) with a local minimum leads to the following local version of our framework's equilibrium. A point $(x^\star, y^\star) \in \mathcal{X} \times \mathcal{Y}$ is a local equilibrium if, for some $\nu > 0$ (and denoting the ball of radius ν at $x^\star$ by $B(x^\star, \nu)$),

$$x^\star \in \operatorname{argmin}_{x\in B(x^\star,\nu)\cap\mathcal{X}}(\max_{y\in P(x,y^*)} f(x, y)), \tag{4}$$

$$y^\star \in \operatorname{argmax}_{y\in P(x^\star,y^\star)} f(x^\star, y), \tag{5}$$

Approximate local equilibrium. Similar to previous work on local minimization for smooth non-convex objectives, we would like to solve (4) and (5) to converge to approximate stationary points [46]. Towards this end, we can replace $P(x, y^\star)$ in (4) and (5) with the set $P_\varepsilon(x, y^\star)$ of endpoints of paths starting at $y^\star$ along which $f(x, \cdot)$ increases at some "rate" $\varepsilon > 0$.

Definition 3.1. *For any $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $\varepsilon \geq 0$, define $P_\varepsilon(x, y)\subseteq\mathcal{Y}$ to be points $w \in \mathcal{Y}$ s.t. there is a continuous and (except at finitely many points) differentiable path $\gamma(t)$ starting at y, ending at w, and unit-speed, i.e., $\left\|\frac{\mathrm{d}}{\mathrm{d}t}\gamma(t)\right\| \leq 1$, s.t. at any point on γ,*

$$\frac{\mathrm{d}}{\mathrm{d}t} f(x, \gamma(t)) \geq \varepsilon.$$

The above definition restricts the max-player to updating y via any "greedy" algorithm, e.g. gradient ascent. Note that, compared to Definition 3.1, the notion of greedy paths in [37] additionally requires

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2} f(x, \gamma(t)) \geq \sqrt{\varepsilon}$$

so as to achieve the goal of converging to an *approximate second-order local equilibrium*. Our goal, on the other hand, is for the max-player's updates to approximate paths taken by first-order greedy algorithms, hence, the condition on first derivative in Definition 3.1 suffices.

While we would also like to replace the local minimum in (4) with an approximate stationary point, the *min-player's objective*, $\mathcal{L}_\varepsilon(x,y) := \max_{z\in P_\varepsilon(x,y)} f(x,z)$, may not be continuous[1] in x, and thus, gradient-based notions of approximate local minimum do not apply. To bypass this difficulty and to define a notion of approximate local minimum which applies to discontinuous functions, we sample updates to x, and test whether $\mathcal{L}_\varepsilon(\cdot,y)$ has decreased. Formally, given a choice of sampling distribution Q_x (which may depend on x), and $\delta,\omega>0$, $x^\star$ is said to be an approximate local minimum of a (possibly discontinuous) function $g:\mathcal{X}\to\mathbb{R}$ if $\Pr_{\Delta\sim Q_{x^\star}}[g(x^\star+\Delta)<g(x^\star)-\delta]<\omega$.

Thus, replacing the set P with P_ε in (4) and (5), and the "exact" local minimum in (4) with an approximate local minimum, we arrive at our equilibrium definition:

Definition 3.2. *Given $\varepsilon,\delta,\omega>0$ and a distribution $Q_{x,y}$, we say that a point $(x^\star,y^\star)\in\mathcal{X}\times\mathcal{Y}$ is an $(\varepsilon,\delta,\omega,Q)$-approximate local equilibrium for our framework if*

$$\Pr_{\Delta\sim Q_{x^\star,y^\star}}\left[\max_{y\in P_\varepsilon(x^\star+\Delta,y^\star)} f(x^\star+\Delta,y) \quad < \max_{y\in P_\varepsilon(x^\star,y^\star)} f(x^\star,y)-\delta\right]\le\omega, \tag{6}$$

$$y^\star\in \operatorname{argmax}_{y\in P_\varepsilon(x^\star,y^\star)} f(x^\star,y). \tag{7}$$

Proposal distribution. In the special case when f is, e.g., $O(1)$-strongly convex in x and $O(1)$-strongly concave in y with $O(1)$-Lipschitz gradients, if one chooses the updates Q to be the (deterministic or stochastic) gradients $-\nabla_x f$, then the $(\varepsilon,\delta,\omega,Q)$-equilibrium corresponds to an "approximate" global min-max point for f with duality gap $O(\varepsilon+\delta)$ (Theorem A.2). This duality gap does not depend on the dimension. This motivates choosing the proposal distribution to be the (stochastic) gradients $-\nabla_x f$ in the more general setting when f is nonconvex-nonconcave. Another motivation for this choice of Q is that adding stochastic gradient noise to steps taken by deep learning algorithms is known empirically to lead to better outcomes than, e.g., standard Gaussian noise (see [58]). Empirically, this choice of Q leads to GANs that are able to successfully learn the dataset's distribution (Section 5).

Comparison to local min-max points. Note that any local min-max point, if it exists, satisfies Definition 3.2 for a proposal distribution Q with small enough mean and variance. This is because if $(x^\star,y^\star)$ is a local min-max point of f, $x^\star$ is a local minimum of $f(\cdot,y^\star)$ and hence there is a ball B containing $x^\star$ on which $x^\star$ minimizes $f(\cdot,y^\star)$. Moreover, $y^\star$ is a first-order stationary point of $f(x^\star,\cdot)$, which means that $P_\varepsilon(x^\star,y^\star)=\{y^\star\}$ and hence

$$\max_{y\in P_\varepsilon(x^\star,y^\star)} f(x^\star,y)=f(x^\star,y^\star),$$

satisfying (7). Therefore, if Q has mean and variance small enough that the min-player's proposed updates $\Delta\sim Q_{x^\star,y^\star}$ fall inside B w.h.p., we will have that

$$\max_{y\in P_\varepsilon(x^\star+\Delta,y^\star)} f(x^\star+\Delta,y)>f(x^\star,y^\star)$$

[1]Consider the example $f(x,y)=\min(x^2y^2,1)$. The min-player's objective for $\varepsilon>0$ is $\mathcal{L}_\varepsilon(x,y)=f(x,y)$ if $2x^2y<\varepsilon$, and 1 otherwise. Thus $\mathcal{L}_{1/2}$ is discontinuous at $(1/2,1)$.

Algorithm 1 Our algorithm for min-max optimization

input: Stochastic zeroth-order oracle F for bounded loss function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with L-Lipschitz gradient, stochastic gradient oracle G_y with mean $\nabla_y f$, Initial point (x_0, y_0)
input: A distribution $Q_{x,y}$, and an oracle for sampling from this distribution. Error parameters $\varepsilon, \delta > 0$
hyperparameters: $\eta > 0$ (learning rate), $r_{\max}$ (maximum number of rejections); τ_1 (for annealing);

Set $i \leftarrow 0$, $r \leftarrow 0$, $\varepsilon_0 = \frac{\varepsilon}{2}$, $f_{\text{old}} \leftarrow \infty$
while $r \leq r_{\max}$ **do**
  Sample Δ_i from the distribution Q_{x_i,y_i}
  Set $X_{i+1} \leftarrow x_i + \Delta_i$ {*min-player's proposed update*}
  Run Algorithm 2 with inputs $\mathsf{x} \leftarrow X_{i+1}$, $\mathsf{y}_0 \leftarrow y_i$, and $\varepsilon' \leftarrow \varepsilon_i \times (1 - 2\eta L)^{-1}$ {*max-player's update*}
  Set $\mathcal{Y}_{i+1} \leftarrow \mathsf{y}_{\text{stationary}}$ to be the output of Algo 2.
  Set $f_{\text{new}} \leftarrow F(X_{i+1}, \mathcal{Y}_{i+1})$ {*Compute the new loss*}
  Set $\mathsf{Accept}_i \leftarrow \mathsf{True}$.
  if $f_{\text{new}} > f_{\text{old}} - \frac{\delta}{4}$, **then**
    Set $\mathsf{Accept}_i \leftarrow \mathsf{False}$ with probability $\max(0, 1 - e^{-\frac{i}{\tau_1}})$ {*Decide to accept or reject*}
  if $\mathsf{Accept}_i = \mathsf{True}$ **then**
    Set $x_{i+1} \leftarrow X_{i+1}$, $y_{i+1} \leftarrow \mathcal{Y}_{i+1}$ {*accept the proposed x and y updates*}
    Set $f_{\text{old}} \leftarrow f_{\text{new}}$, $r \leftarrow 0$, $\varepsilon_{i+1} \leftarrow \varepsilon_i \times (1 - 2\eta L)^{-2}$
  else
    Set $x_{i+1} \leftarrow x_i$, $y_{i+1} \leftarrow y_i$, $r \leftarrow r + 1$, $\varepsilon_{i+1} \leftarrow \varepsilon_i$ {*Reject the proposed updates*}
  Set $i \leftarrow i + 1$
return $(x^\star, y^\star) \leftarrow (x_i, y_i)$

since $y^\star \in P_\varepsilon(x^\star + \Delta, y^\star)$, implying that $(x^\star, y^\star)$ satisfies (6) (proof provided in Appendix C).

However, the converse is not true. This is a necessary feature of our definition as there are simple smooth bounded functions which do not have any local min-max points and yet an equilibrium from Definition 3.2 is guaranteed to exist. For instance, as mentioned earlier, $\sin(x+y)$ does not have any local min-max points; however, the global min-max points $S = \{(x, y) : x + y = \frac{\pi}{2} + 2\pi k, k \in \mathbb{N}\}$ of $\sin(x+y)$ satisfy Definition 3.2 for any $\varepsilon > 0$, $\delta = \Omega(\sqrt{\varepsilon})$, $\omega = 0$, and, e.g., any proposal distribution Q with support on a ball of radius $\frac{1}{2}$ centered at 0. (See Appendix B for examples)

3.2 Algorithm

We present an algorithm for our framework (Algorithm 1), along with the gradient ascent subroutine it uses to compute max-player updates (Algorithm 2). In Theorem 3.3, we show it efficiently finds an approximate local equilibrium (Definition 3.2). We consider bounded loss functions $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, where f is an empirical risk loss over m training examples, i.e.,

$$f := \frac{1}{m} \sum_{i \in [m]} f_i.$$

We assume we are given access to f via a randomized oracle F where $\mathbb{E}[F] = f$. We call such an oracle a stochastic zeroth-order oracle for f. We are also given randomized oracles G_x, G_y for $\nabla_x f$, $\nabla_y f$, where $\mathbb{E}[G_x] = \nabla_x f$, and $\mathbb{E}[G_y] = \nabla_y f$, and call such oracles stochastic gradient oracles for f. These oracles are computed by randomly sampling "batches" $B, B_x, B_y \subseteq [m]$ (iid, with

replacement) and returning

$$F = \frac{1}{|B|}\sum_{i\in B} f_i, G_x = \frac{1}{|B_x|}\sum_{i\in B_x} \nabla_x f_i, \text{ and } G_y = \frac{1}{|B_y|}\sum_{i\in B_y} \nabla_y f_i.$$

For our convergence guarantees, we require the following bounds on standard smoothness parameters for each $f_i : b, L > 0$ such that $|f_i(x, y)| \leq b$ and

$$\|\nabla f_i(x, y) - \nabla f_i(x', y')\| \leq L\|x - x'\| + L\|y - y'\|$$

for all x, y and all i. These bounds imply f is also b-bounded, and L-gradient-Lipschitz.

Algorithm 2 Stochastic gradient ascent (for max-player updates)

input: Stochastic gradient oracle G_y for $\nabla_y f$; initial points x, y_0; error parameter ε'
hyperparameters: $\eta > 0$

Set $j \leftarrow 0$, $\mathsf{Stop} \leftarrow$ False
while $\mathsf{Stop} =$ False **do**
  Set $g_{\mathrm{y},j} \leftarrow G_y(\mathsf{x}, \mathsf{y}_j)$
  if $\|g_{\mathrm{y},j}\| > \varepsilon'$ **then**
    Set $\mathsf{y}_{j+1} \leftarrow \mathsf{y}_j + \eta g_{\mathrm{y},j}$,
    Set $j \leftarrow j + 1$
  else
    Set $\mathsf{Stop} \leftarrow$ True
return $\mathsf{y}_{\text{stationary}} \leftarrow \mathsf{y}_j$

Overview and intuition of our algorithm. From the current point (x, y), Algorithm 1 first proposes a random update Δ from the given distribution $Q_{x,y}$ to update the min-player's parameters to $x + \Delta$. In practice, we oftentimes choose $Q_{x,y}$ to be the distribution of the (scaled) stochastic gradient $-G_x$, although one may implement Algorithm 1 with any choice of distribution $Q_{x,y}$. Then, it updates the max-player's parameters greedily by running gradient ascent using the stochastic gradients G_y until it reaches a first-order ε-stationary point y', that is, a point where

$$\|\nabla_y f(x + \Delta, y')\| \leq \varepsilon.$$

Thus, the point y' satisfies (7). However, Algorithm 1 still needs to eventually find a pair of points $(x^\star, y^\star)$ where $x^\star$ is an approximate local minimum of the min-player's objective $\mathcal{L}_\varepsilon(\cdot, y^\star)$ in order to satisfy (6). Moreover, it must also ensure that $y^\star$ satisfies (7). Towards this, Algorithm 1 does the following:

1) The algorithm re-uses this same point y' to compute an approximation $f(x + \Delta, y')$ for $\mathcal{L}_\varepsilon(x + \Delta, y)$ in order to have access to the value of the min-player's objective $\mathcal{L}_\varepsilon$ to be able to minimize it.

2) If $f(x + \Delta, y')$ is less than $f(x, y)$ the algorithm concludes that $\mathcal{L}_\varepsilon(x + \Delta, y)$ has decreased and, consequently, accepts the updates $x + \Delta$ and y'; otherwise it rejects both updates. We show that after accepting both $x + \Delta$ and y', $\mathcal{L}_\varepsilon(x + \Delta, y') < \mathcal{L}_\varepsilon(x, y)$, implying that the algorithm does not cycle.

3) It then starts the next iteration proposing a random update which again depends on its current position.

4) While Algorithm 1 does not cycle, to avoid getting stuck, if it is unable to decrease $\mathcal{L}_\varepsilon$ after roughly $1/\omega$ attempts, it concludes w.h.p. that the current x is an approximate local minimum for $\mathcal{L}_\varepsilon(\cdot, y)$ with respect to the given distribution. This is because, by definition, at an approximate local minimum, a random update from the given distribution has probability at most ω of decreasing $\mathcal{L}_\varepsilon$. We also show that the current y is an ε-stationary point for $f(x, \cdot)$.

We conclude this section with a few remarks: 1) In practice our algorithm can be implemented just as easily with ADAM instead of SGD, as in some of our experiments (alternately, one may also be able to substitute other optimization algorithms such as Momentum SGD [48], ADAGrad [15], or Adabelief [59] for gradient updates). 2) Algorithm 1 uses a randomized accept-reject rule (similar to simulated annealing)– if the resulting loss has decreased, the updates for x and y are accepted; otherwise they are only accepted with a small probability e^{-i/τ_1} at each iteration i, where τ_1 is a "temperature" parameter. 3) While our main result still holds if one replaces simulated annealing with a deterministic acceptance rule, the annealing step seems to be beneficial in practice in the early period of training when our algorithm is implemented with ADAM gradients. 4) Finally, in simulations, we find that Algorithm 1's implementation can be simplified by taking a small fixed number of max-player updates at each iteration.

3.3 Convergence Guarantee

Theorem 3.3 (Main result). *Algorithm 1, with hyperparameters $\eta > 0$, $\tau_1 > 0$, given access to stochastic zeroth-order and gradient oracles for a function $f = \sum_{i\in[m]} f_i$ where each f_i is b-bounded with L-Lipschitz gradient for some $b, L > 0$, and $\varepsilon, \delta, \omega > 0$, and an oracle for sampling from a distribution $Q_{x,y}$, with probability at least $9/10$ returns $(x^\star, y^\star) \in \mathbb{R}^d \times \mathbb{R}^d$ such that, for some $\varepsilon^\star \in [0.5\varepsilon, \varepsilon]$, $(x^\star, y^\star)$ is an $(\varepsilon^\star, \delta, \omega, Q)$-approximate local equilibrium. The number of stochastic gradient, function, and sampling oracle calls required by the algorithm is* $\mathrm{poly}\,(b, L, 1/\varepsilon, 1/\delta, 1/\omega)$ *and does not depend on the dimension d.*

Theorem 3.3 says that our algorithm is guaranteed to converge to an approximate local equilibrium for our framework from any starting point, for any f which is bounded with Lipschitz gradients including nonconvex-nonconcave f. As discussed in related work this is in contrast to prior works which assume e.g., that $f(x, y)$ is concave in y or that the algorithm is provided with an initial point such that the underlying continuous dynamics converge to a local min-max point. The exact number of stochastic gradient, function, and sampling oracle calls required by the algorithm is $\tilde{O}(b^3L^3/(\delta^3\omega^3\varepsilon^4))$. We present a proof overview for Theorem 3.3 next, and the full proof in Section 6.

In the setting where $f(x, y)$ is α-strongly convex in x and α-strongly concave in y and has L-Lipschitz gradients, Algorithm 3.3 with x-player updates $Q_{x,y}$ chosen to be the x-gradients $-\nabla_x f(x, y)$, outputs a point $(x^\star, y^\star)$ which is an $(\varepsilon, \varepsilon, 0.25, Q)$-equilibrium point in $\mathrm{poly}(1/\varepsilon, L, 1/\alpha, D)$ gradient evaluations, where $D := \|(x_0, y_0) - (x^\dagger, y^\dagger)\|$ is the distance from the initial point to the global min-max point $(x^\dagger, y^\dagger)$. As mentioned in Section 1, since f is strongly convex-strongly concave with Lipschitz gradients, this point is also an approximate global min-max point with duality gap $O(\varepsilon)$, and the number of gradient evaluations for our algorithm to achieve a duality gap $O(\varepsilon)$ is independent of the dimension (Corollary A.10).

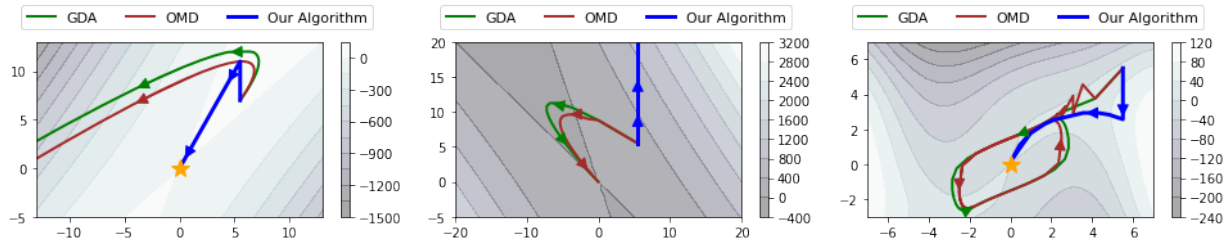

Figure 2: Our algorithm (blue), GDA (green), and OMD (red) on test functions F_1 (left), F_2 (center), and F_3 (right). F_1 and F_3 have global min-max points at (0,0) (yellow star), and F_2 has no min-max points since $\min_{x\in\mathbb{R}} \max_{y\in\mathbb{R}} F_2(x,y) = +\infty$.

4 Proof Overview for Theorem 3.3

For simplicity, assume $b = L = \tau_1 = 1$ and $\varepsilon = \delta = \omega$. There are two key pieces to proving Theorem 3.3. The first is to show that our algorithm converges to some point $(x^\star, y^\star)$ in a number of gradient, function, and sampling oracle calls that is $\text{poly}(1/\varepsilon)$ and independent of the dimension d (Lemma 4.1). Secondly, we show that, $y^\star$ is a first-order ε-stationary point for $f(x^\star, \cdot)$, and $x^\star$ is an approximate local minimum of $\mathcal{L}_\varepsilon(\cdot, y^\star)$ (Lemma 4.2).

Step 1: Bounding the number of oracle evaluations.

Lemma 4.1 (Informal, see Lemma 6.5). *Algorithm 1 terminates after at most* $\text{poly}(b, L, 1/\varepsilon, 1/\delta, 1/\omega)$ *gradient, function, and sampling oracle evaluations.*

Proof outline of Lemma 4.1. After $\Theta(\log(1/\varepsilon))$ iterations of Algorithm 1, the decaying acceptance rate (Line 10 of Algorithm 1) ensures that, with probability at least $1 - O(\varepsilon)$, at any iteration i for which Algorithm 1 accepts a proposed update to (x_i, y_i), we have that

$$f(x_{i+1}, y_{i+1}) \le f(x_i, y_i) - \varepsilon. \tag{8}$$

Next, we note that the stopping condition in Line 2 of Algorithm 1 implies our algorithm stops whenever $r_{\max} = \Theta(1/\varepsilon)$ proposed steps are rejected in a row. Thus, (8) implies that for every $\Theta(r_{\max})$ iterations where the algorithm does not terminate, with probability at least $1 - O(\varepsilon)$ the value of the loss decreases by at least $\Omega(\varepsilon)$. Since f is 1-bounded, this implies our algorithm terminates after roughly $O(r_{\max}/\varepsilon)$ iterations of the minimization routine w.h.p. (Prop. 6.4).

Next, we use the fact that $G_y(x, y)$ is a batch gradient,

$$G_y(x,y) = \frac{1}{|B_y|} \sum_{i \in B_y} \nabla_y f_i(x,y),$$

of batch size $|B_y| = O(1/\varepsilon^2 \log(1/\varepsilon))$, together with the Azuma–Hoeffding concentration inequality, to show w.h.p. that

$$\|G_y(x,y) - \nabla_y f(x,y)\| \le O(\varepsilon), \tag{9}$$

(Proposition 6.1). We then use (9) together with the fact f is 1-bounded with 1-Lipschitz gradient, to show that, w.h.p., the maximization subroutine (Algorithm 2) requires at most $\text{poly}(1/\varepsilon)$ stochastic gradient ascent steps to reach an ε-stationary point (Proposition 6.3). As each step of the max-subroutine requires one gradient evaluation, and each iteration of the min-routine calls the max-subroutine once (and makes O(1) oracle calls), the total number of oracle calls is $\text{poly}(1/\varepsilon)$. □

Step 2: Show $x^\star$ is approximate local minimum for $\mathcal{L}_\varepsilon(\cdot, y^\star)$, and $y^\star$ is ε-stationary point.

Lemma 4.2 (Informal, see Lemma 6.7). *W.h.p., the output $(x^\star, y^\star)$ of Algorithm 1 is an approximate local equilibrium for our framework, for parameters $(\varepsilon, \delta, \omega)$ and proposal distribution Q.*

Proof outline of Lemma 4.2. Since we have already shown that Algorithm 2 runs stochastic gradient ascent until it reaches a ε-stationary point, $\|\nabla_{\mathrm{y}} f(x^\star, y^\star)\| \le \varepsilon$. The accept/reject rule (Line 10 of Algorithm 1) says that the proposed update $x^\star + \Delta$ is rejected with probability at least $1 - O(\varepsilon)$ whenever

$$f(x^\star + \Delta, y') \ge f(x^\star, y^\star) - \varepsilon, \tag{10}$$

where the maximization subroutine computes y' by gradient ascent on $f(x^\star + \Delta, \cdot)$ initialized at $y^\star$. And the stopping condition in Line 2 of Algorithm 1 implies that the last $r_{\max}$ updates $x^\star + \Delta$ proposed by the min-player were all rejected, and hence were sampled from the distribution $Q_{x^\star, y^\star}$. Roughly, this fact together with (10) implies that, with high probability, the proposal distribution $Q_{x^\star, y^\star}$ at the point $(x^\star, y^\star)$ satisfies

$$\begin{aligned} \Pr_{\Delta \sim Q_{x^\star, y^\star}} [f(x^\star + \Delta, y') \ge f(x^\star, y^\star) - \varepsilon] &\ge 1 - O(r_{\max}^{-1}) \\ &= 1 - O(\varepsilon). \end{aligned} \tag{11}$$

To show (6) holds, we need to replace f in the above equation with the min-player's objective $\mathcal{L}_\varepsilon$. Towards this end, we first use the fact that f has $O(1)$-Lipschitz gradient, together with (9), to show that, w.h.p., the stochastic gradient ascent steps of Algorithm 2 form an "ε-increasing" path, starting at $y^\star$ with endpoint y', along which f increases at rate at least ε (Prop. 6.6). Since $\mathcal{L}_\varepsilon$ is the supremum of f at the endpoints of *all* such ε-increasing paths starting at $y^\star$,

$$f(x^\star + \Delta, y') \le \mathcal{L}_\varepsilon(x^\star + \Delta, y^\star). \tag{12}$$

Finally, recall (Section 3) that $\|\nabla_{\mathrm{y}} f(x^\star, y^\star)\| \le \varepsilon^\star$ implies that $\mathcal{L}_\varepsilon(x^\star, y^\star) = f(x^\star, y^\star)$, and hence (7) holds. Plugging this and (12) into (11) implies that

$$\Pr_{\Delta \sim Q_{x^\star, y^\star}} \left[\mathcal{L}_\varepsilon(x^\star + \Delta, y') \ge \mathcal{L}_\varepsilon(x^\star, y^\star) - \varepsilon\right] \ge 1 - O(\varepsilon),$$

and hence that (6) holds. □

5 Empirical Results

5.1 Performance on Test Functions

We apply our algorithm to three test loss functions previously considered in [56]:

$$F_1(x, y) = -3x^2 - y^2 + 4xy,$$

$$F_2(x, y) = 3x^2 + y^2 + 4xy,$$

$$F_3(x, y) = (4x^2 - (y - 3x + 0.05x^3)^2 - 0.1y^4)e^{-0.01(x^2+y^2)}.$$

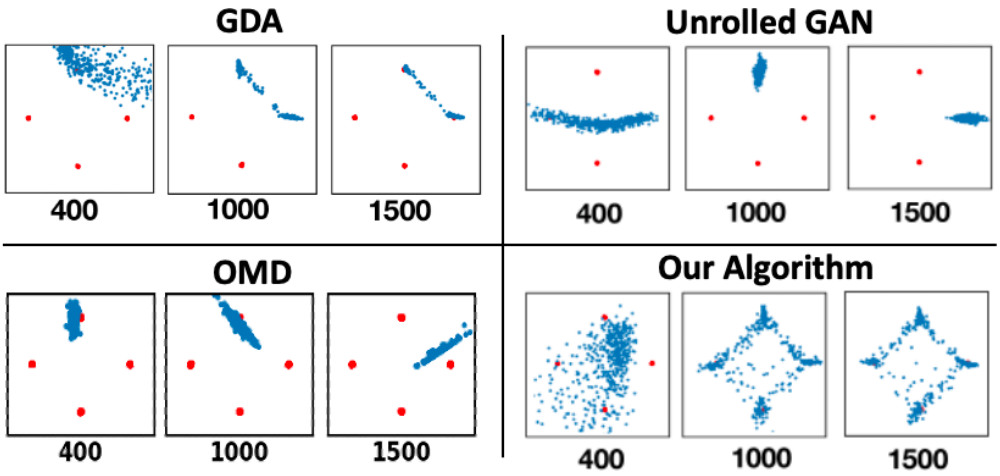

Figure 3: Our algorithm, unrolled GANs with $k = 6$ unrolling steps, OMD, and GDA with $k = 6$ max-player steps trained on a 4-Gaussian mixture for 1500 iterations. Our algorithm used $k = 6$ max-player steps and acceptance rate $e^{-1/\tau} = 0.25$. Plots show the points generated by each algorithm after the specified iterations.

We choose these functions because they are known to be challenging for gradient-based algorithms.

Both F_1 and F_3 have global min-max at $(0, 0)$, yet popular gradient-based algorithms including GDA, OMD, and extra-gradient (EG) algorithm were shown in [56] not to converge on these functions. In contrast, we observe that our algorithm finds the global min-max points of both F_1 and F_3. To see why our algorithm converges, note that it uses the maximization subroutine (Algorithm 2) to first find the "ridge" along which $f(x, y)$ is a local maximum in the y variable, and to then return to a point on the ridge every time the min-player proposes an update. Since the min-player in our algorithm only accepts updates which lead to a net decrease in f, our algorithm eventually finds the point $(0, 0)$ on this ridge where f is minimized. In comparison, for GDA and OMD, the max-player's gradient $\nabla_y f$ is zero along the ridge where $f(x, y)$ is a local maximum in the y variable, while the min-player's gradient $-\nabla_x f$ can be large; on F_1 and F_3 $-\nabla_x f$ points away from this ridge, and this can prevent GDA and OMD from converging to the point $(0, 0)$.

In case of F_2, $\min_{x\in\mathbb{R}} \max_{y\in\mathbb{R}} f(x, y) = +\infty$. On F_2, GDA, OMD, and EG all *converge* to $(0, 0)$ which is neither a global min-max nor a local min-max point. In contrast, our algorithm diverges to infinity.

When applying our algorithm we use $\eta = 0.05$ and $Q_{x,y} \sim N(0, 0.25)$. For GDA and OMD we use learning rate 0.05 (see Appendix D.1).

5.2 Performance when Training GANs

We apply our algorithm to train GANs to learn from both synthetic and real-world datasets. When training on both datasets, we choose the proposal distribution Q in our algorithm to be the (ADAM) stochastic gradients for $-\nabla_x f$. We formulate GAN using our framework with cross entropy loss,

$$f(x, y) = -\left(\log(\mathcal{D}_y(\zeta)) + \log(1 - \mathcal{D}_y(\mathcal{G}_x(\xi)))\right),$$

where x, y are the parameters of generator $\mathcal{G}$ and discriminator $\mathcal{D}$ respectively, ζ is sampled from data, and $\xi \sim N(0, I_d)$.

Table 1: Gaussian mixture dataset. The fraction of times (out of 20 runs) each method generates m modes, for $m \in [4]$. k is the number of max-player steps per iteration. Our algorithm learns 4 modes in more runs than other algorithms.

	Number of modes learnt			
Method	1	2	3	4
This paper	0	0.15	0.15	**0.70**
GDA ($k = 1$)	0.95	0.05	0	0
GDA ($k = 6$)	0.05	0.75	0	0.20
OMD	0.80	0.20	0	0
Unrolled-GAN	0.75	0.15	0.10	0

To adapt Algorithm 1 to training GANs, we make certain simplifications: 1) we use a fixed temperature τ at all iterations i, making it simpler to choose a good temperature value, rather than a temperature schedule; 2) we replace the randomized acceptance rule with a deterministic rule: If $f_{\text{new}} \leq f_{\text{old}}$ we accept, and if $f_{\text{new}} > f_{\text{old}}$ we only accept if i is a multiple of $e^{\frac{1}{\tau}}$ (i.e., average acceptance rate of $e^{-\frac{1}{\tau}}$); 3) we take a fixed number of max-player steps at each iteration, instead of taking as many steps as needed to achieve a small gradient. These simplifications do not significantly affect our algorithm's performance (see Appendix G).

Gaussian mixture dataset. This synthetic dataset consists of 512 points sampled from a mixture of four equally weighted Gaussians in two dimensions with standard deviation 0.01 and means at $(0, 1)$, $(1, 0)$, $(-1, 0)$, $(0, -1)$. Since modes in this dataset are well-separated, mode collapse can be clearly detected. We report the number of modes learnt by the GAN from each training algorithm across iterations.

Baselines. We compare our algorithm's performance to GDA, OMD [13], and unrolled GAN [40]. For the networks and hyperparameter details, see Appendix D.3.

Results. We trained GANs on the Gaussian mixture dataset for 1500 iterations using our algorithm, unrolled GANs with 6 unrolling steps, GDA with $k = 1$ and $k = 6$ max-player steps (using Adam updates), and OMD with $k = 6$ max-player steps. We repeated each simulation 20 times. The performance of the output GAN learned by all algorithms is presented in Table 1, while Figure 3 shows the samples from generators of the different training algorithms at various iterations (see Appendix D.4 for images from all runs). The GAN returned by our algorithm learns all four modes in 70% of the runs, significantly more than the other training algorithms (Table 1). Thus, for this synthetic dataset, our algorithm is the most effective in avoiding mode collapse and cycling in comparison to baselines.

Results on real-world datasets. While we focus on 2-D and Gaussian mixture GAN simulations in this section to illustrate the convergence properties of our algorithm, we also ran our algorithm on two real-world datasets, 01-MNIST and CIFAR-10. For the 01-MNIST dataset, samples generated from GANs trained using GDA and our algorithm are presented in Figure 4. We observed that GANs trained on the 01-MNIST dataset with the gradient descent-ascent algorithm (GDA) exhibit

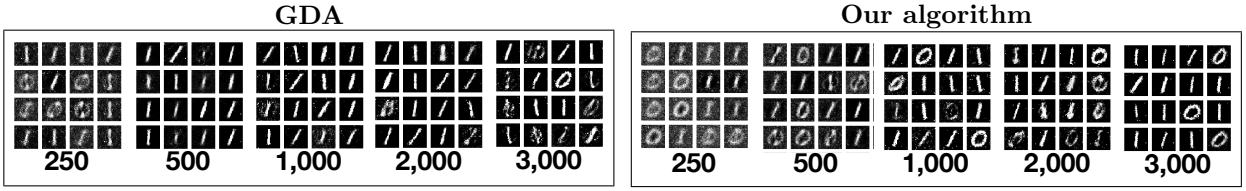

Figure 4: Images generated at various iterations by the GAN trained using GDA vs our algorithm (for 3000 iterations) on the 01-MNIST dataset. See Appendix F for more results and details.

Table 2: CIFAR-10 dataset: The mean (and standard error) of Inception Scores of models from different training algorithms. Note that, GDA and our algorithm return generators with similar mean performance; however, the standard error of the Inception Score in case of GDA is relatively larger.

	Iteration		
Method	5000	25000	50000
Ours	2.71 (0.28)	4.10 (0.35)	**4.68** (0.39)
GDA	2.80 (0.52)	4.28 (0.77)	4.51 (0.86)
OMD	1.60 (0.18)	1.73 (0.25)	1.96 (0.26)

mode collapse in 77% of the trial runs (Figure 19, in Appendix F), while GANs trained with our algorithm do not exhibit mode collapse in any of the training runs (Figure 20 in Appendix F). For the CIFAR-10 dataset, samples generated from GANs trained using our algorithm are presented in Figure 5. On CIFAR-10, our algorithm achieved a mean Inception score of 4.68 after 50k iterations (across 20 repetitions); in comparison, GDA achieved a mean Inception score of 4.51 and OMD achieved a mean Inception score of 1.96 (Table 2). Detailed results and methodologies used for 01-MNIST and CIFAR-10 datasets are presented in Appendix F and E respectively.

Experiments with GANs also demonstrate that our algorithm scales to high-dimensional parameter spaces; the dimension d of the space of trainable parameters used in the GAN experiments was around 3.5×10^4 for the GANs trained on the Gaussian mixture dataset, 3×10^6 for 01-MNIST and 2×10^6 for CIFAR-10. [2]

6 Proof of Theorem 3.3

In this section we give the proof of Theorem 3.3.

Setting parameters: We start by setting parameters which will be used in the proof. Let $\mathfrak{b}_0 = |B|$, $\mathfrak{b}_y = |B_y|$ denote the batch sizes. And note that the fact that each f_i has L-Lipschitz gradient for all $i \in [m]$, implies that each f_i is also L_1-Lipschitz, where $L_1 = \sqrt{2Lb}$.

For the theoretical analysis, we assume $0 < \varepsilon \leq 1$, and set the following parameters:

[2] The code for the above simulations is available at `https://github.com/vijaykeswani/Min-Max-Optimization-Algorithm`.

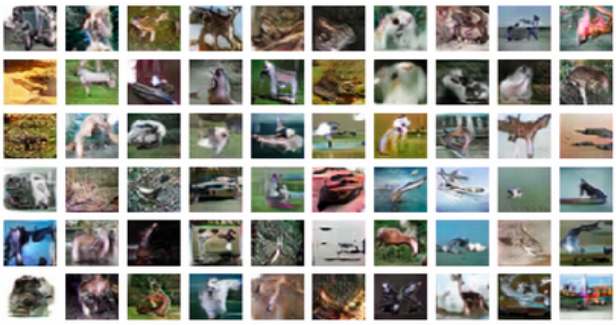

Figure 5: Images generated by the GAN trained using our algorithm on the CIFAR-10 dataset. See Appendix E for more results and samples from GANs trained using other baselines.

1. $\nu = \frac{1}{20}\left[\frac{320b(L+1)}{\varepsilon^2}\left(\tau_1 \log\left(\frac{128}{\omega^2}\right) + \frac{2048b}{\omega\delta}\log^2\left(\frac{100}{\omega}\left(\tau_1+1\right)\left(8\frac{b}{\delta}+1\right)\right)+1\right)\right]^{-2}$
2. $r_{\max} = \frac{128}{\omega}\log^2\left(\frac{100}{\omega}(\tau_1+1)\left(8\frac{b}{\delta}+1\right) + \log\left(\frac{1}{\nu}\right)\right)$
3. Define $\mathcal{I} := \tau_1 \log(\frac{r_{\max}}{\nu}) + 8r_{\max}\frac{b}{\delta} + 1$
4. $\eta = \min\left(\frac{1}{10L}, \frac{1}{8L\mathcal{I}}\right)$
5. Define $\mathcal{J} := \frac{16b}{\eta\varepsilon^2}$
6. $\hat{\varepsilon}_1 = \min\left(\varepsilon\eta L, \frac{\delta}{8}\right)$
7. $\mathfrak{b}_0 = \hat{\varepsilon}_1^{-2} 300^2 b^2 \log(\frac{1}{\nu})$
8. $\mathfrak{b}_y = \varepsilon^{-2}\hat{\varepsilon}_1^{-2} 300^2 L_1^2 \log(\frac{1}{\nu})$

In particular, note that

$$\nu \leq \frac{1}{20}\left(2\mathcal{J}\mathcal{I} + 2 \times \left(r_{\max}\frac{8b}{\delta}+1\right)\right)^{-1} \text{ and } r_{\max} \geq \frac{4}{\omega}\log\left(\frac{100\mathcal{I}}{\omega}\right).$$

At every iteration $i \leq \mathcal{I}$, where we set $\varepsilon' = \varepsilon_i$. We also have

$$\varepsilon' \leq \varepsilon_0\left(\frac{1}{1-2\eta L}\right)^{2i} \leq \varepsilon. \tag{13}$$

To see why (13) holds, note that since we set the hyperparameter η to be $\eta = \min\left(\frac{1}{10L}, \frac{1}{8L\mathcal{I}}\right)$, we have

$$1 - 2\eta L \leq 1 - \frac{1}{4\mathcal{I}}.$$

Since we also set $\varepsilon_0 = \frac{\varepsilon}{2}$, we therefore have that for all $i \leq \mathcal{I}$,

$$\varepsilon_0(1-2\eta L)^{-2i} \leq \frac{\varepsilon}{2}\left(1-\frac{1}{4\mathcal{I}}\right)^{-2\mathcal{I}} \leq \varepsilon,$$

where the second inequality holds because

$$\left(1 - \frac{1}{2t}\right)^{-t} \leq 2$$

for all $t \geq 1$.

6.1 Step 1: Bounding the Number of Gradient, Function, and Sampling Oracle Evaluations

The first step in our proof is to bound the number of gradient, function, and sampling oracle evaluations required by our algorithm. Towards this end, we begin by showing a concentration bound (Proposition 6.1) for the value of the stochastic gradient and function oracles used by our algorithm. Next, we bound the number of iterations of its discriminator update subroutine Algorithm 2 (Proposition (17)), and the number of iterations in Algorithm 1 (Proposition 6.4); together, these two bounds imply a poly$(b, L, 1/\varepsilon, 1/\delta, 1/\omega)$ bound on the number of gradient, function, and sampling oracle evaluations (Lemma 6.5).

Proposition 6.1. *For any $\hat{\varepsilon}_1, \nu > 0$, if we use batch sizes $\mathfrak{b}_y = \varepsilon^{-2}\hat{\varepsilon}_1^{-2}300^2L_1^2\log(1/\nu)$ and $\mathfrak{b}_0 = \hat{\varepsilon}_1^{-2}300^2b^2\log(1/\nu)$, we have that*

$$\mathbb{P}\left(\|G_y(x,y) - \nabla_y f(x,y)\| \geq \frac{\hat{\varepsilon}_1}{10}\right) < \nu, \tag{14}$$

and

$$\mathbb{P}\left(|F(x,y) - f(x,y)| \geq \frac{\hat{\varepsilon}_1}{10}\right) < \nu. \tag{15}$$

Proof. From Section 3 we have that

$$G_y(x,y) - \nabla_y f(x,y) = \frac{1}{\mathfrak{b}_y}\sum_{i \in B_y}[\nabla_y f_i(x,y) - \nabla_y f(x,y)],$$

where the batch $B_y \subseteq [m]$ is sampled iid with replacement from $[m]$. But since each f_i has L-Lipschitz gradient, we have (with probability 1) that

$$\|\nabla_y f_i(x,y) - \nabla_y f(x,y)\| \leq \|\nabla_y f_i(x,y)\| + \|\nabla_y f(x,y)\| \leq 2L_1.$$

Now,

$$\mathbb{E}[\nabla_y f_i(x,y) - \nabla_y f(x,y)] = \mathbb{E}[\nabla_y f_i(x,y) - \mathbb{E}[\nabla_y f_i(x,y)]] = 0.$$

Therefore, by the Azuma–Hoeffding inequality for mean-zero bounded vectors, we have

$$\mathbb{P}\left(\left\|\frac{1}{\mathfrak{b}_y}\sum_{i \in B_y}[\nabla_y f_i(x,y) - \nabla_y f(x,y)]\right\| \geq \frac{s\sqrt{\mathfrak{b}_y}+1}{\mathfrak{b}_y}2L_1\right) < 2e^{1-\frac{1}{2}s^2} \qquad \forall s > 0.$$

Hence, if we set $s = 6\log^{\frac{1}{2}}\left(\frac{2}{\nu}\right)$, we have that

$$7\log^{\frac{1}{2}}\left(\frac{2}{\nu}\right)\sqrt{\mathfrak{b}_y}+1 \geq s\sqrt{\mathfrak{b}_y}+1$$

and hence

$$\mathbb{P}\left(\left\|\frac{1}{\mathfrak{b}_y}\sum_{i\in B_y}[\nabla_y f_i(x,y)-\nabla_y f(x,y)]\right\| \geq \frac{7\log^{\frac{1}{2}}(\frac{2}{\nu})\sqrt{\mathfrak{b}_y}}{\mathfrak{b}_y}2L_1\right) < \nu.$$

Therefore,

$$\mathbb{P}\left(\left\|\frac{1}{\mathfrak{b}_y}\sum_{i\in B_y}[\nabla_y f_i(x,y)-\nabla_y f(x,y)]\right\| \geq \frac{\hat{\varepsilon}_1}{10}\right) < \nu$$

which completes the proof of Inequality (14). Inequality (15) follows from the exact same steps as the proof of Inequality (14), if we replace the bound L_1 for $\|\nabla_y f_i(x,y)\|$ with the bound b on $|f_i(x,y)|$.

□

Proposition 6.2. *For every j, with probability at least $1-\nu$ we have that either $\|G_y(\mathsf{x},\mathsf{y}_j)\| < \varepsilon$, or that*

$$\|\nabla_y f(\mathsf{x},\mathsf{y}_j) - G_y(\mathsf{x},\mathsf{y}_j)\| \leq \frac{1}{10}\eta L \times \min\left(\|G_y(\mathsf{x},\mathsf{y}_j)\|,\ \|\nabla_y f(\mathsf{x},\mathsf{y}_j)\| + \frac{\hat{\varepsilon}_1}{10}\right) \tag{16}$$

and

$$\|\mathsf{y}_{j+1}-\mathsf{y}_j\| = \eta\|G_y(\mathsf{x},\mathsf{y}_j)\| \leq 2\eta\|\nabla_y f(\mathsf{x},\mathsf{y}_j)\| \tag{17}$$

Proof. By Proposition 6.1, we have that, with probability at least $1-\nu$, and whenever $\|G_y(\mathsf{x},\mathsf{y}_j)\| \geq \varepsilon$,

$$\|\nabla_y f(\mathsf{x},\mathsf{y}_j) - G_y(\mathsf{x},\mathsf{y}_j)\| < \frac{\hat{\varepsilon}_1}{10} \leq \frac{1}{10}\varepsilon\eta L \leq \frac{1}{10}\eta L \times \min\left(\|G_y(\mathsf{x},\mathsf{y}_j)\|,\ \|\nabla_y f(\mathsf{x},\mathsf{y}_j)\| + \frac{\hat{\varepsilon}_1}{10}\right),$$

where the first inequality holds by Proposition 6.1, the second inequality holds since $\hat{\varepsilon}_1 \leq \varepsilon\eta L$, and the third inequality holds since $\|G_y(\mathsf{x},\mathsf{y}_j)\| \geq \varepsilon$ and since (again by Proposition 6.1)

$$\|\nabla_y f(\mathsf{x},\mathsf{y}_j) - G_y(\mathsf{x},\mathsf{y}_j)\| < \frac{\hat{\varepsilon}_1}{10}.$$

This proves Inequality (16). Moreover, we have that, whenever $\|G_y(\mathsf{x},\mathsf{y}_j)\| \geq \varepsilon$ and in the same probability $1-\nu$ event where (16) holds,

$$\|\eta G_y(\mathsf{x},\mathsf{y}_j)\| \leq \eta\left(\|\nabla_y f(\mathsf{x},\mathsf{y}_j)\| + \frac{\hat{\varepsilon}_1}{10}\right). \tag{18}$$

Thus,

$$2\eta\|\nabla_y f(\mathsf{x},\mathsf{y}_j)\| \geq 2\eta\left(\|G_y(\mathsf{x},\mathsf{y}_j)\| - \frac{\hat{\varepsilon}_1}{10}\right) \geq \eta\|G_y(\mathsf{x},\mathsf{y}_j)\| = \|\mathsf{y}_{j+1}-\mathsf{y}_j\|,$$

where the first inequality holds by (18), the second inequality holds since $\|G_y(\mathsf{x},\mathsf{y}_j)\| \geq \varepsilon \geq \hat{\varepsilon}_1$, and the equality holds by Step 5 of Algorithm 2. This proves Inequality (17). □

Proposition 6.3. *Algorithm 2 terminates in at most* $\mathcal{J} := \frac{16b}{\eta\varepsilon^2}$ *iterations of its "While" loop, with probability at least* $1 - \nu \times \mathcal{J}$.

Proof. Let $j_{\max} \in \mathbb{N} \cup \{\infty\}$ be the number of iterations of the "While" loop in Algorithm 2. First, we note that the stopping condition for Algorithm 2 implies that

$$\|G_y(\mathsf{x}, \mathsf{y}_j)\| \geq \frac{1}{2}\varepsilon \tag{19}$$

for all $j \leq j_{\max} - 1$. Since f has L-Lipschitz gradient, there exits a vector u, with $\|u\| \leq L\|\mathsf{y}_{j+1} - \mathsf{y}_j\|$, such that, for all $j \leq j_{\max} - 1$

$$\begin{aligned}
f(\mathsf{y}_{j+1}) - f(\mathsf{y}_j) &= \langle \mathsf{y}_{j+1} - \mathsf{y}_j,\ \nabla_y f(\mathsf{x}, \mathsf{y}_j) + u\rangle \qquad (20)\\
&= \langle \mathsf{y}_{j+1} - \mathsf{y}_j,\ \nabla_y f(\mathsf{x}, \mathsf{y}_j)\rangle + \langle \mathsf{y}_{j+1} - \mathsf{y}_j,\ u\rangle\\
&= \langle \eta G_y(\mathsf{x}, \mathsf{y}_j),\ G_y(\mathsf{x}, \mathsf{y}_j)\rangle - \langle \eta G_y(\mathsf{x}, \mathsf{y}_j),\ G_y(\mathsf{x}, \mathsf{y}_j) - \nabla_y f(\mathsf{x}, \mathsf{y}_j)\rangle + \langle \eta G_y(\mathsf{x}, \mathsf{y}_j),\ u\rangle\\
&\geq \eta\|G_y(\mathsf{x}, \mathsf{y}_j)\|^2 - \eta\|G_y(\mathsf{x}, \mathsf{y}_j)\| \times \|G_y(\mathsf{x}, \mathsf{y}_j) - \nabla_y f(\mathsf{x}, \mathsf{y}_j)\| - \eta\|G_y(\mathsf{x}, \mathsf{y}_j)\| \times \|u\|\\
&\overset{\text{Prop.6.2}}{\geq} \eta\|G_y(\mathsf{x}, \mathsf{y}_j)\|^2 - \eta\|G_y(\mathsf{x}, \mathsf{y}_j)\| \times \frac{\eta L}{10}\|G_y(\mathsf{x}, \mathsf{y}_j)\| - \eta\|G_y(\mathsf{x}, \mathsf{y}_j)\| \times L\|\mathsf{y}_{j+1} - \mathsf{y}_j\|\\
&= \eta\|G_y(\mathsf{x}, \mathsf{y}_j)\|^2 - \frac{1}{10}\eta^2 L\|G_y(\mathsf{x}, \mathsf{y}_j)\|^2 - \eta\|G_y(\mathsf{x}, \mathsf{y}_j)\| \times L\|\eta G_y(\mathsf{x}, \mathsf{y}_j)\|\\
&\geq \frac{1}{8}\eta\|G_y(\mathsf{x}, \mathsf{y}_j)\|^2\\
&\overset{\text{Eq.19}}{\geq} \frac{1}{2}\eta\varepsilon^2,
\end{aligned}$$

with probability at least $1 - \nu$, where the second-to-last inequality holds since $\eta \leq \frac{1}{10L}$. Existence of the vector u in Equation (20) is guaranteed by the fundamental theorem of calculus. Namely, by the fundamental theorem of calculus we have

$$f(\mathsf{y}_{j+1}) - f(\mathsf{y}_j) = \int_0^1 \langle \mathsf{y}_{j+1} - \mathsf{y}_j, \nabla_y f(\mathsf{x}, \mathsf{y}_j + t(\mathsf{y}_{j+1} - \mathsf{y}_j))\rangle \mathrm{d}t.$$

Thus (20) holds for $u = \int_0^1 \nabla_y f(\mathsf{x}, \mathsf{y}_j + t(\mathsf{y}_{j+1} - \mathsf{y}_j)) - \nabla_y f(\mathsf{x}, \mathsf{y}_j)\mathrm{d}t$. Note that this choice of u satisfies $\|u\| \leq L\|\mathsf{y}_{j+1} - \mathsf{y}_j\|$, since $\|\nabla_y f(\mathsf{x}, \mathsf{y}_j + t(\mathsf{y}_{j+1} - \mathsf{y}_j)) - \nabla_y f(\mathsf{x}, \mathsf{y}_j)\| \leq L\|\mathsf{y}_{j+1} - \mathsf{y}_j\|$ for $t \in [0, 1]$ because f has L-Lipschitz gradient. Since f takes values in $[-b, b]$, Inequality (20) implies that Algorithm 2 terminates in at most $\mathcal{J} = \frac{16b}{\eta\varepsilon^2}$ iterations of its "While" loop, with probability at least $1 - \nu \times \mathcal{J}$.

□

Proposition 6.4. *Algorithm 1 terminates in at most* $\mathcal{I} := \tau_1 \log(\frac{r_{\max}}{\nu}) + 8r_{\max}\frac{b}{\delta} + 1$ *iterations of its "While" loop, with probability at least* $1 - 2\nu \times (r_{\max}\frac{8b}{\delta} + 1)$.

Proof. For any $i > 0$, let E_i be the "bad" event that both $f(x_{i+1}, y_{i+1}) - f(x_i, y_i) > -\frac{\delta}{4}$ and $\mathsf{Accept}_i = \mathsf{True}$. Then by Proposition 6.1, since $\frac{\hat{\varepsilon}_1}{10} \leq \frac{\delta}{8}$, we have that

$$\mathbb{P}(E_i) \leq e^{-\frac{i}{\tau_1}} + \nu. \tag{21}$$

Define $\hat{\mathcal{I}} := \tau_1 \log(\frac{r_{\max}}{\nu})$. Then for $i \geq \hat{\mathcal{I}}$, from Line 10 of Algorithm 1 we have by Inequality (21) that

$$\mathbb{P}(E_i) \leq 2\nu.$$

Define $h := r_{\max}\frac{8b}{\delta} + 1$. Then

$$\mathbb{P}\left(\bigcup_{i=\hat{\mathcal{I}}}^{\hat{\mathcal{I}}+h} E_i\right) \leq 2\nu \times h. \tag{22}$$

Since f takes values in $[-b, b]$, if $\bigcup_{i=\hat{\mathcal{I}}}^{\hat{\mathcal{I}}+h} E_i$ does not occur, the number of accepted steps over the iterations $\hat{\mathcal{I}} \leq i \leq \hat{\mathcal{I}} + h$ (that is, the size of the set $\{i : \hat{\mathcal{I}} \leq i \leq \hat{\mathcal{I}} + h, \mathsf{Accept}_i = \mathsf{True}\}$) is at most $\frac{8b}{\delta}$.

Therefore, since $h = r_{\max}\frac{8b}{\delta} + 1$, we must have that there exists a number $\mathfrak{i}$, with $\hat{\mathcal{I}} \leq \mathfrak{i} \leq \mathfrak{i} + r_{\max} \leq \hat{\mathcal{I}} + h$, such that $\mathsf{Accept}_i = \mathsf{False}$ for all $i \in [\mathfrak{i}, \mathfrak{i} + r_{\max}]$. Therefore the condition in the While loop (Line 2) of Algorithm 1 implies that Algorithm 1 terminates after at most $\mathfrak{i} + r_{\max} \leq \hat{\mathcal{I}} + h$ iterations of its While loop, as long as $\bigcup_{i=\hat{\mathcal{I}}}^{\hat{\mathcal{I}}+h} E_i$ does not occur. Therefore, Inequality (22) implies that, with probability at least $1 - 2\nu \times \left(r_{\max}\frac{8b}{\delta} + 1\right)$, Algorithm 1 terminates after at most

$$\hat{\mathcal{I}} + h = \tau_1 \log\left(\frac{r_{\max}}{\nu}\right) + 8 r_{\max}\frac{b}{\delta} + 1$$

iterations of its "While" loop.

□

Lemma 6.5. *With probability at least $1 - 3\nu\mathcal{J}\mathcal{I}$, Algorithm 1 terminates after at most $(\tau_1 \log(\frac{r_{\max}}{\nu}) + 4 r_{\max}\frac{b}{\delta} + 1) \times (\mathcal{J} \times \mathfrak{b}_y + \mathfrak{b}_0 + \mathfrak{b}_x)$ gradient, function, and sampling oracle evaluations.*

Proof. Each iteration of the While loop in Algorithm 1 computes one batch gradient with batch size $\mathfrak{b}_x$, one stochastic function evaluation of batch size $\mathfrak{b}_0$, generates one sample from the proposal distribution Q, and calls Algorithm 2 exactly once. Each iteration of the While loop in Algorithm 2 computes one batch gradient with batch size $\mathfrak{b}_y$. The result then follows directly from Propositions 6.4 and 6.3. □

6.2 Step 2: Proving the Output $(x^\star, y^\star)$ of Algorithm 1 is an Approximate Local Equilibrium

The second step in our proof is to show that the output of Algorithm 1 is an approximate local equilibrium (Definition 3.2) for our framework with respect to $\varepsilon, \delta, \omega > 0$ and the distribution $Q_{x,y}$ (Lemma 6.7). Towards this end, we first show that the steps taken by the discriminator update subroutine (Algorithm 2) form a path along which the loss f is increasing (Proposition 6.6).

Recall the paths $\gamma(t)$ from Definition 3.1. From now on we will refer to such paths as "ε-increasing paths". That is, for any $\varepsilon > 0$, we say that a path $\gamma(t)$ is an "ε-increasing path" if at every point along this path we have that $\left\|\frac{\mathrm{d}}{\mathrm{d}t}\gamma(t)\right\| = 1$ and that $\frac{\mathrm{d}}{\mathrm{d}t} f(x, \gamma(t)) \geq \varepsilon$.

Proposition 6.6. *Every time Algorithm 2 is called we have that, with probability at least $1 - 2\nu\mathcal{J}$, the path consisting of the line segments $[\mathsf{y}_j, \mathsf{y}_{j+1}]$ formed by the points y_j computed by Algorithm 2 has a parametrization $\gamma(t)$ which is an $(1 - 2\eta L)\varepsilon'$-increasing path.*

Proof. We consider the following continuous unit-speed parametrized path $\gamma(t)$:

$$\gamma(t) = \mathsf{y}_j + \left(t - \sum_{k=1}^{j-1} \|v_k\|\right) \frac{v_j}{\|v_j\|}, \qquad \forall\ t \in \left[\sum_{k=1}^{j-1} \|v_k\|, \sum_{k=1}^{j} \|v_k\|\right], \quad j \in [j_{\max}],$$

where $v_j = \eta G_y(\mathsf{x}, \mathsf{y}_j)$ and $j_{\max}$ is the number of iterations of the While loop of Algorithm 2. Next, we show that $\frac{\mathrm{d}}{\mathrm{d}t} f(\mathsf{x}, \gamma(t)) \geq \varepsilon'$. For each $j \in [j_{\max}]$ we have that

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} f(\mathsf{x}, \gamma(t)) &\geq [\nabla_y f(\mathsf{x}, \mathsf{y}_j) - L\|\mathsf{y}_{j+1} - \mathsf{y}_j\| u]^\top \frac{v_j}{\|v_j\|} \\
&= [\nabla_y f(\mathsf{x}, \mathsf{y}_j) - L\eta\|G_y(\mathsf{x}, \mathsf{y}_j)\| u]^\top \frac{v_j}{\|v_j\|} \\
&\overset{\text{Prop.6.1}}{\geq} \left[G_y(\mathsf{x}, \mathsf{y}_j) - \frac{1}{10}\eta L \|G_y(\mathsf{x}, \mathsf{y}_j)\| w - L\eta \|G_y(\mathsf{x}, \mathsf{y}_j)\| u\right]^\top \frac{G_y(\mathsf{x}, \mathsf{y}_j)}{\|G_y(\mathsf{x}, \mathsf{y}_j)\|} \\
&\geq \|G_y(\mathsf{x}, \mathsf{y}_j)\| - \frac{1}{10}\eta L \|G_y(\mathsf{x}, \mathsf{y}_j)\| - L\eta \|G_y(\mathsf{x}, \mathsf{y}_j)\| \\
&\geq (1 - 2\eta L)\|G_y(\mathsf{x}, \mathsf{y}_j)\| \\
&\geq (1 - 2\eta L)\varepsilon' \qquad\qquad \forall t \in \left[\sum_{k=1}^{j-1} \|v_k\|, \sum_{k=1}^{j} \|v_k\|\right],
\end{aligned} \tag{23}$$

with probability at least $1 - \nu$ for some unit vectors $u, w \in \mathbb{R}^d$. But by Proposition 6.3 we have that $j_{\max} \leq \mathcal{J}$ with probability at least $1 - \nu \times \mathcal{J}$. Therefore inequality (23) implies that

$$\frac{\mathrm{d}}{\mathrm{d}t} f(\mathsf{x}, \gamma(t)) \geq (1 - 2\eta L)\varepsilon' \qquad \forall t \in \left[0, \sum_{k=1}^{j_{\max}} \|v_k\|\right],$$

with probability at least $1 - 2\nu\mathcal{J}$.

□

Lemma 6.7. *Let $i^\star$ be such that $i^\star - 1$ is the last iteration i of the "While" loop in Algorithm 1 for which $\mathsf{Accept}_i = \mathsf{True}$. Then with probability at least $1 - 2\nu\mathcal{J}\mathcal{I} - 2\nu \times \left(r_{\max}\frac{8b}{\delta} + 1\right)$ we have that*

$$\|\nabla_y f(x^\star, y^\star)\| \leq (1 - \eta L)\varepsilon_{i^\star}. \tag{24}$$

Moreover, with probability at least $1 - \frac{\omega}{100} - 2\nu \times \left(r_{\max}\frac{8b}{\delta} + 1\right)$ we have that

$$\mathbb{P}_{\Delta \sim Q_{x^\star, y^\star}}\left(\mathcal{L}_{\varepsilon_{i^\star}}(x^\star + \Delta, y^\star) \leq \mathcal{L}_{\varepsilon_{i^\star}}(x^\star, y^\star) - \frac{1}{2}\delta \middle| x^\star, y^\star\right) \leq \frac{1}{2}\omega. \tag{25}$$

and that

$$\frac{\varepsilon}{2} \leq \varepsilon_{i^\star} \leq \varepsilon. \tag{26}$$

Proof. First, we note that $(x^\star, y^\star) = (x_i, y_i)$ for all $i \in \{i^\star, \ldots, i^\star + r_{\max}\}$, and that Algorithm 1 stops after exactly $i^\star + r_{\max}$ iterations of the "While" loop in Algorithm 1.

Let H_i be the "bad" event that, when Algorithm 2 is called during the ith iteration of the "While" loop in Algorithm 1, the path traced by Algorithm 2 is not an ε_i-increasing path. Then, by Proposition 6.6 we have that

$$\mathbb{P}(\mathsf{H}_i) \leq 2\nu\mathcal{J}. \tag{27}$$

Let K_i be the "bad" event that $\|G_y(x_i, y_i) - \nabla_y f(x_i, y_i)\| \geq \frac{\hat{\varepsilon}_1}{10}$. Then by Propositions 6.1 and 6.3 we have that

$$\mathbb{P}(\mathsf{K}_i) \leq 2\nu\mathcal{J}. \tag{28}$$

Whenever K_i^c occurs we have that

$$\begin{aligned} \|\nabla_y f(x_i, y_i)\| &\leq \|G_y(x_i, y_i)\| + \|G_y(x_i, y_i) - \nabla_y f(x_i, y_i)\| \\ &\leq (1 - 2\eta L)\varepsilon_i + \|G_y(x_i, y_i) - \nabla_y f(x_i, y_i)\| \\ &\leq (1 - 2\eta L)\varepsilon_i + \frac{\hat{\varepsilon}_1}{10} \\ &\leq (1 - \eta L)\varepsilon_i, \end{aligned} \tag{29}$$

where the second Inequality holds by Line 4 of Algorithm 2, and the last inequality holds since $\frac{\hat{\varepsilon}_1}{10} \leq \eta L$. Therefore, Inequalities (28) and (29) together with Proposition 6.4 imply that

$$\|\nabla_y f(x^\star, y^\star)\| \leq (1 - \eta L)\varepsilon_{i^\star}$$

with probability at least $1 - 2\nu\mathcal{J}\mathcal{I} - 2\nu \times \left(r_{\max}\frac{8b}{\delta} + 1\right)$. This proves Inequality (24).

Inequality (29) also implies that, whenever K_i^c occurs, the set $P_{\varepsilon_i}(x_i, y_i)$ of endpoints of ε_i-increasing paths with initial point y_i (and x-value x_i) consists only of the single point y_i. Therefore, we have that

$$\mathcal{L}_{\varepsilon_i}(x_i, y_i) = f(x_i, y_i) \tag{30}$$

whenever K_i^c occurs. Moreover, whenver H_i^c occurs we have that $\mathcal{Y}_{i+1}$ is the endpoint of an ε_i-increasing path with starting point $(x_i + \Delta_i, y_i)$. Now, $\mathcal{L}_{\varepsilon_i}(x_i + \Delta_i, y_i)$ is the supremum of the value of f at the endpoints of all ε_i-increasing paths with starting point $(x_i + \Delta_i, y_i)$. Therefore, we must have that

$$\mathcal{L}_{\varepsilon_i}(x_i + \Delta_i, y_i) \geq f(x_i + \Delta_i, \mathcal{Y}_{i+1}) \tag{31}$$

whenever H_i^c occurs. Therefore,

$$\begin{aligned} &\mathbb{P}_{\Delta\sim Q_{x_i,y_i}}\left(\mathcal{L}_{\varepsilon_i}(x_i + \Delta, y_i) > \mathcal{L}_{\varepsilon_i}(x_i, y_i) - \frac{1}{2}\delta\middle| x_i, y_i\right) \\ &\overset{\text{Eq.30,31}}{\geq} \mathbb{P}_{\Delta\sim Q_{x_i,y_i}}\left(f(x_i + \Delta, \mathcal{Y}_{i+1}) > f(x_i, y_i) - \frac{1}{2}\delta\middle| x_i, y_i\right) - \mathbb{P}(\mathsf{H}_i) - \mathbb{P}(\mathsf{K}_i) \\ &\overset{\text{Prop.6.1}}{\geq} \mathbb{P}_{\Delta\sim Q_{x_i,y_i}}\left(F(x_i + \Delta, \mathcal{Y}_{i+1}) > F(x_i, y_i) - \frac{1}{4}\delta\middle| x_i, y_i\right) - 2\nu - \mathbb{P}(\mathsf{H}_i) - \mathbb{P}(\mathsf{K}_i) \\ &\geq \mathbb{P}\left(\mathsf{Accept}_i = \mathsf{False}\middle| x_i, y_i\right) - 2\nu - \mathbb{P}(\mathsf{H}_i) - \mathbb{P}(\mathsf{K}_i) \end{aligned} \tag{32}$$

$$\overset{\text{Eq.27,28}}{\geq} \mathbb{P}\left(\mathsf{Accept}_i = \mathsf{False}\middle| x_i, y_i\right) - 2\nu - 2\nu\mathcal{J} - 2\nu\mathcal{J}, \;\; \forall i \leq \mathcal{I},$$

where the second inequality holds by Proposition 6.1, since $\frac{\hat{\varepsilon}_1}{10} \leq \frac{\delta}{8}$. Define

$$p_i := \mathbb{P}_{\Delta \sim Q_{x_i, y_i}}\left(\mathcal{L}_{\varepsilon_i}(x_i + \Delta, y_i) > \mathcal{L}_{\varepsilon_i}(x_i, y_i) - \frac{1}{2}\delta \middle| x_i, y_i\right)$$

for every $i \in \mathbb{N}$. Then Inequality (32) implies that

$$\mathbb{P}\left(\mathsf{Accept}_i = \mathsf{False}\middle| x_i, y_i\right) \leq p_i + \nu(4\mathcal{J} + 2) \leq p_i + \frac{1}{8}\omega \qquad \forall i \leq \mathcal{I}, \tag{33}$$

since $\nu \leq \omega/(32\mathcal{J} + 16)$. We now consider what happens for indices i for which $p_i \leq 1 - \omega/2$. Since $(x_{i+s}, y_{i+s}) = (x_i, y_i)$ whenever $\mathsf{Accept}_{i+k} = \mathsf{False}$ for all $0 \leq k \leq s$, we have by Inequality (33) that

$$\mathbb{P}\left(\cap_{s=0}^{r_{\max}}\{\mathsf{Accept}_{i+s} = \mathsf{False}\}\middle| p_i \leq 1 - \frac{1}{2}\omega\right) \leq \left(1 - \frac{1}{4}\omega\right)^{r_{\max}} \leq \frac{\omega}{100\mathcal{I}} \qquad \forall i \leq \mathcal{I} - r_{\max}$$

since

$$r_{\max} \geq \frac{4}{\omega}\log\left(\frac{100\mathcal{I}}{\omega}\right).$$

Therefore, with probability at least $1 - \frac{\omega}{100\mathcal{I}} \times \mathcal{I} = 1 - \frac{\omega}{100}$, we have that the event $\cap_{s=0}^{r_{\max}}\{\mathsf{Accept}_{i+s} = \mathsf{False}\}$ does not occur for any $i \leq \mathcal{I} - r_{\max}$ for which $p_i \leq 1 - \frac{1}{2}\omega$. Recall from Proposition 6.4 that Algorithm 1 terminates in at most $\mathcal{I}$ iterations of its "While" loop, with probability at least $1 - 2\nu \times \left(r_{\max}\frac{8b}{\delta} + 1\right)$. Therefore,

$$\mathbb{P}\left(p_{i^\star} > 1 - \frac{1}{2}\omega\right) \geq 1 - \frac{\omega}{100} - 2\nu \times \left(r_{\max}\frac{8b}{\delta} + 1\right). \tag{34}$$

In other words, by the definition of $p_{i^\star}$, Inequality (34) implies that with probability at least $1 - \frac{\omega}{100} - 2\nu \times \left(r_{\max}\frac{8b}{\delta} + 1\right)$, the point $(x^\star, y^\star)$ is such that

$$\mathbb{P}_{\Delta \sim Q_{x^\star, y^\star}}\left(\mathcal{L}_{\varepsilon_{i^\star}}(x^\star + \Delta, y^\star) \leq \mathcal{L}_{\varepsilon_{i^\star}}(x^\star, y^\star) - \frac{1}{2}\delta \middle| x^\star, y^\star\right) \leq \frac{1}{2}\omega.$$

This completes the proof of inequality (25). Finally we note that when Algorithm 1 terminates in at most $\mathcal{I}$ iterations of its "While" loop, we have

$$\varepsilon_{i^\star} = \varepsilon_0\left(\frac{1}{1 - 2\eta L}\right)^{2i^\star} \leq \varepsilon_0\left(\frac{1}{1 - 2\eta L}\right)^{2\mathcal{I}} \leq \varepsilon, \tag{35}$$

since $\eta \leq \frac{1}{8L\mathcal{I}}$. This completes the proof of Inequality (26). □

We can now complete the proof of the main theorem:

Proof of Theorem 3.3. First, by Lemma 6.5, with probability at least $1 - 3\nu\mathcal{J}\mathcal{I} \geq 99/100$, our algorithm converges to some point $(x^\star, y^\star)$ after at most

$$\left(\tau_1 \log\left(\frac{r_{\max}}{\nu}\right) + 4r_{\max}\frac{b}{\delta} + 1\right) \times (\mathcal{J} \times \mathfrak{b}_y + \mathfrak{b}_0 + \mathfrak{b}_x)$$

gradient, function, and sampling oracle evaluations, which is polynomial in $b, L_1, L, 1/\varepsilon, 1/\delta, 1/\omega$, and does not depend on the dimension d. By Lemma 6.7, if we set $\varepsilon^\star = \varepsilon_{i^\star}$, we have that Inequalities (7) and (6) hold for parameters $\varepsilon^\star \in \left[\frac{1}{2}\varepsilon, \varepsilon\right]$, δ, ω and distribution Q, with probability at least

$$1 - 2\nu\mathcal{J}\mathcal{I} - 2\nu \times \left(r_{\max}\frac{8b}{\delta} + 1\right) \geq \frac{19}{20},$$

since $\nu \leq \frac{1}{20}\left(2\mathcal{J}\mathcal{I} + 2 \times \left(r_{\max}\frac{8b}{\delta} + 1\right)\right)^{-1}$. □

7 Conclusion and Future Directions

We introduce a new variant of the min-max optimization framework, and provide a gradient-based algorithm with efficient convergence guarantees to an equilibrium for this framework, for nonconvex-nonconcave losses and from any initial point. Empirically, we observe our algorithm converges on many challenging test functions and shows improved stability when training GANs.

While we show our algorithm runs in time polynomial in b, L, and independent of dimension d, we do not believe our bounds are tight and it would be interesting to show the run-time is linear in b, L. Moreover, while our guarantees hold for any distribution Q, it would be interesting to see if a specialized analysis for adaptively preconditioned distributions can lead to improved bounds. Our framework can also be extended to more general settings; e.g., to multi-agent minimization problems arising in meta-learning [19].

Acknowledgments

This research was supported in part by NSF CCF-1908347, NSF CCF-2112665, and NSF CCF-2104528 grants and an AWS ML research award.

References

[1] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1195–1199, 2017.

[2] Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations, ICLR*, 2017.

[3] Rowel Atienza. GAN by example using Keras on Tensorflow backend. "https://towardsdatascience.com/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0", 2017.

[4] David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 354–363. PMLR, 2018.

[5] Ali Borji. Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[6] Jason Brownlee. How to develop a GAN to generate CIFAR10 small color photographs. "https://machinelearningmastery.com/how-to-develop-a-generative-adversarial-network-for-a-cifar-10-small-object-photographs-from-scratch/", 2019.

[7] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8, 2017.

[8] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. In *International Conference on Learning Representations, ICLR*, 2017.

[9] Cong D Dang and Guanghui Lan. On the convergence properties of non-euclidean extragradient methods for variational inequalities with generalized monotone operators. *Computational Optimization and applications*, 60(2):277–310, 2015.

[10] John M Danskin. The theory of max-min, with applications. *SIAM Journal on Applied Mathematics*, 14(4):641–664, 1966.

[11] Constantinos Daskalakis and Ioannis Panageas. The limit points of (optimistic) gradient descent in min-max optimization. In *Advances in Neural Information Processing Systems 31*, pages 9236–9246. Curran Associates, Inc., 2018.

[12] Constantinos Daskalakis and Ioannis Panageas. Last-iterate convergence: Zero-sum games and constrained min-max optimization. In *10th Innovations in Theoretical Computer Science Conference, ITCS*, pages 27:1–27:18, 2019.

[13] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training GANs with optimism. In *International Conference on Learning Representations*, 2018.

[14] Jelena Diakonikolas, Constantinos Daskalakis, and Michael Jordan. Efficient methods for structured nonconvex-nonconcave min-max optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2746–2754. PMLR, 2021.

[15] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[16] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martín Arjovsky, Olivier Mastropietro, and Aaron C. Courville. Adversarially learned inference. In *5th International Conference on Learning Representations, ICLR*, 2017.

[17] Tanner Fiez, Benjamin Chasnov, and Lillian J Ratliff. Convergence of learning dynamics in Stackelberg games. *arXiv preprint arXiv:1906.01217*, 2019.

[18] Tanner Fiez, Benjamin Chasnov, and Lillian Ratliff. Implicit learning dynamics in stackelberg games: Equilibria characterization, convergence analysis, and empirical study. In *International Conference on Machine Learning*, pages 3133–3144. PMLR, 2020.

[19] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[20] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pages 797–842, 2015.

[21] Gauthier Gidel, Reyhane Askari Hemmat, Mohammad Pezeshki, Rémi Le Priol, Gabriel Huang, Simon Lacoste-Julien, and Ioannis Mitliagkas. Negative momentum for improved game dynamics. In *Proceedings of Machine Learning Research*, volume 89, pages 1802–1811. PMLR, 2019.

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[24] Chi Jin, Praneeth Netrapalli, and Michael Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *International Conference on Machine Learning*, pages 4880–4889. PMLR, 2020.

[25] Renu Khandelwal. Generative adversarial network (GAN) using Keras. "https://medium.com/datadriveninvestor/generative-adversarial-network-gan-using-keras-ce1c05cfdfd3", 2019.

[26] David Kinderlehrer and Guido Stampacchia. An introduction to variational inequalities and their applications. *Bull. Amer. Math. Soc*, 7:622–627, 1982.

[27] Weiwei Kong and Renato DC Monteiro. An accelerated inexact proximal point method for solving nonconvex-concave min-max problems. *SIAM Journal on Optimization*, 31(4):2558–2585, 2021.

[28] GM Korpelevich. The extragradient method for finding saddle points and other problems. *Matekon: translations of Russian and East European mathematical economics*, 12:747–756, 1976.

[29] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[30] Jerry Li, Aleksander Madry, John Peebles, and Ludwig Schmidt. On the limitations of first-order approximation in GAN dynamics. In *International Conference on Machine Learning*, pages 3011–3019, 2018.

[31] Tengyuan Liang and James Stokes. Interaction matters: A note on non-asymptotic local convergence of generative adversarial networks. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 907–915, 2019.

[32] Tianyi Lin, Chi Jin, and Michael Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *International Conference on Machine Learning*, pages 6083–6093. PMLR, 2020.

[33] Mingrui Liu, Youssef Mroueh, Jerret Ross, Wei Zhang, Xiaodong Cui, Payel Das, and Tianbao Yang. Towards better understanding of adaptive gradient algorithms in generative adversarial nets. In *International Conference on Learning Representations*, 2019.

[34] Mingrui Liu, Hassan Rafique, Qihang Lin, and Tianbao Yang. First-order convergence theory for weakly-convex-weakly-concave min-max problems. *Journal of Machine Learning Research*, 22(169):1–34, 2021.

[35] Songtao Lu, Ioannis Tsaknakis, Mingyi Hong, and Yongxin Chen. Hybrid block successive approximation for one-sided non-convex min-max problems: algorithms and applications. *IEEE Transactions on Signal Processing*, 2020.

[36] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations, ICLR*, 2018.

[37] Oren Mangoubi and Nisheeth K. Vishnoi. Greedy adversarial equilibrium: An efficient alternative to nonconvex-nonconcave min-max optimization. In *ACM Symposium on Theory of Computing (STOC)*, 2021.

[38] Panayotis Mertikopoulos, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile. In *International Conference on Learning Representations, ICLR*, 2019.

[39] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of GANs. In *Advances in Neural Information Processing Systems*, pages 1825–1835, 2017.

[40] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *5th International Conference on Learning Representations, ICLR*, 2017.

[41] Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil. A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach. In *International Conference on Artificial Intelligence and Statistics*, pages 1497–1507. PMLR, 2020.

[42] Aryan Mokhtari, Asuman E Ozdaglar, and Sarath Pattathil. Convergence rate of $O(1/k)$ for optimistic gradient and extragradient methods in smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 30(4):3230–3251, 2020.

[43] Vaishnavh Nagarajan and J Zico Kolter. Gradient descent GAN optimization is locally stable. In *Advances in Neural Information Processing Systems*, pages 5585–5595, 2017.

[44] Arkadi Nemirovski. Prox-method with rate of convergence $O(1/T)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

[45] Arkadi S Nemirovski and David Berkovich Yudin. Cesari convergence of the gradient method of approximating saddle points of convex-concave functions. In *Doklady Akademii Nauk*, volume 239, pages 1056–1059. Russian Academy of Sciences, 1978.

[46] Yurii Nesterov and Boris T Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[47] Maher Nouiehed, Maziar Sanjabi, Tianjian Huang, Jason D Lee, and Meisam Razaviyayn. Solving a class of non-convex min-max games using iterative first order methods. In *Advances in Neural Information Processing Systems 32*, pages 14905–14916. 2019.

[48] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

[49] Hassan Rafique, Mingrui Liu, Qihang Lin, and Tianbao Yang. Weakly-convex–concave min–max optimization: provable algorithms and applications in machine learning. *Optimization Methods and Software*, pages 1–35, 2021.

[50] Tim Salimans, Han Zhang, Alec Radford, and Dimitris Metaxas. Improving GANs using optimal transport. In *International Conference on Learning Representations, ICLR*, 2018.

[51] Shibani Santurkar, Ludwig Schmidt, and Aleksander Madry. A classification-based study of covariate shift in GAN distributions. In *International Conference on Machine Learning*, pages 4480–4489. PMLR, 2018.

[52] Chaobing Song, Zhengyuan Zhou, Yichao Zhou, Yong Jiang, and Yi Ma. Optimistic dual extrapolation for coherent non-monotone variational inequalities. *Advances in Neural Information Processing Systems*, 33, 2020.

[53] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in GANs using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318, 2017.

[54] Kiran K Thekumparampil, Prateek Jain, Praneeth Netrapalli, and Sewoong Oh. Efficient algorithms for smooth minimax optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[55] Nisheeth K. Vishnoi. *Algorithms for Convex Optimization*. Cambridge University Press, 2021.

[56] Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. In *International Conference on Learning Representations*, 2019.

[57] Junchi Yang, Negar Kiyavash, and Niao He. Global convergence and variance reduction for a class of nonconvex-nonconcave minimax problems. *Advances in Neural Information Processing Systems*, 33:1153–1165, 2020.

[58] Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *International Conference on Machine Learning*, pages 7654–7663. PMLR, 2019.

[59] Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, 33:18795–18806, 2020.

A Equilibrium in the Strongly Convex-Strongly Concave Setting

In this section we show that, if f is strongly-convex strongly-concave with Lipschitz gradient, if we choose Q to be the distribution of (either deterministic or stochastic) gradients with mean $-\nabla_x f$, then an $(\varepsilon, \delta, \omega, Q)$-equilibrium corresponds to an "aproximate" global min-max point (Theorem A.1). We then show that this fact, together with the proof of our main result, implies that when our algorithm is applied to α-strongly-convex α-strongly-concave objective functions f with L-Lipschitz gradient, it finds an "approximate" global min-max point with duality gap $O(\varepsilon)$ in a number of gradient evaluations that is polynomial in $L, \frac{1}{\varepsilon}, \frac{1}{\alpha}, L, D$ (Corollary A.10).

For any $L > 0$, we say that a function $\psi : \mathbb{R}^d \to \mathbb{R}$ has L-Lipschitz gradient (equivalently, "L-smooth") if for any $x, \theta \in \mathbb{R}^d$,

$$\|\nabla\psi(x) - \nabla\psi(\theta)\| \leq L \times ||x - \theta\|.$$

And for any $\alpha > 0$ we say that ψ is α-strongly convex if for any $x, \theta \in \mathbb{R}^d$,

$$(\nabla\psi(x) - \nabla\psi(\theta))^\top (x - \theta) \geq \alpha\|x - \theta\|^2.$$

Similarly, we say that ψ is α-strongly concave if $-\psi$ is α strongly-convex.

In the following, we assume that the proposal distribution is a stochastic gradient for $-\nabla_x f$ with some variance $\sigma^2 \geq 0$; for simplicity we set $\sigma^2 = 0$ in Corollary A.10, although this is not strictly necessary.

Assumption A.1. ($\sigma \geq 0$) *For every $(x, y) \in \mathbb{R}^d \times \mathbb{R}^d$, the distribution $Q_{x,y}$ satisfies $\mathbb{E}_{\Delta \sim Q_{x,y}}[\Delta] = -\frac{1}{2L}\nabla_x f(x, y)$ and $\mathbb{E}_{\Delta \sim Q_{x,y}}[\| - \frac{1}{2L}\nabla_x f(x, y) - \Delta\|^2] \leq \sigma^2$.*

Theorem A.2. *Suppose that $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is α-strongly convex in x and α-strongly concave in y, with L-Lipschitz gradient in both variables for some $L \geq \alpha > 0$. Then, for any $\varepsilon, \delta, \omega > 0$ with $\omega \leq \frac{1}{2}$, and any proposal distribution $Q_{x,y}$ satisfying Assumption* (A.1) *for some $\sigma \geq 0$, we have that any point $(x^\star, y^\star)$ which is an $(\varepsilon, \delta, \omega, Q)$-approximate local equilibrium of f also has duality gap satisfying*

$$\max_{y \in \mathbb{R}^d} f(x^\star, y) - \min_{x \in \mathbb{R}^d} f(x, y^\star) \leq \frac{L\varepsilon^2}{2\alpha^2} + \frac{L^3}{\alpha^2}\left(\sqrt{\delta + L\left(2\frac{\varepsilon}{\alpha} + \frac{1}{\alpha}(L\frac{\varepsilon}{\alpha} + 2\sigma)\right) + L\frac{\varepsilon^2}{2\alpha^2}} + L\frac{\varepsilon}{\alpha} + L\frac{\varepsilon}{\alpha}\right)^2.$$

Before proving Theorem A.2, we first show a number of Lemmas. In the following we set $\mu = \frac{1}{L}$.

Lemma A.3. *For any $x, w \in \mathbb{R}^d$,*

$$\|\text{argmax}_{z \in \mathbb{R}^d} f(x, z) - \text{argmax}_{z \in \mathbb{R}^d} f(\theta, z)\| \leq \frac{L}{\alpha}\|\theta - x\|$$

Proof. Since $f(x, \cdot)$ is concave, we have that a point z is a global maximum if and only if $\nabla_y f(x, z) = 0$. Since $f(x, \cdot)$ is α-strongly concave for $\alpha > 0$, this global maximum point is unique.

Let $z^\star$ be the global maximum of $f(x, \cdot)$, and let $\zeta^\star$ be the global maximum of $f(\theta, \cdot)$. Then, since $f(x, \cdot)$ is α-strongly concave, $\|\nabla_y f(x, z)\| \geq \alpha\|z - z^\star\|$ for all $z \in \mathbb{R}^d$. Moreover, since f is L-smooth, we also have that $\|\nabla_y f(x, z) - \nabla_y f(\theta, z)\| \leq L\|\theta - x\|$ for every $x, \theta, z \in \mathbb{R}^d$. Therefore,

$$\begin{aligned}\|\nabla_y f(\theta,z)\| &\geq \|\nabla_y f(x,z)\| - \|\nabla_y f(\theta,z) - \nabla_y f(x,z)\| \\ &\geq \|\nabla_y f(x,z)\| - L\|\theta - x\| \\ &\geq \alpha\|z - z^\star\| - L\|\theta - x\|. \end{aligned} \tag{36}$$

Since $\alpha\|z - z^\star\| - L\|\theta - x\| > 0$ for any $z \in \mathbb{R}^d$ such that $\|z - z^\star\| > \frac{L}{\alpha}\|\theta - x\|$, (36) implies that

$$\|\text{argmax}_{z\in\mathbb{R}^d} f(x,z) - \text{argmax}_{z\in\mathbb{R}^d} f(\theta,z)\| \leq \frac{L}{\alpha}\|\theta - x\|$$

□

Lemma A.4. *If $(x,w) \in \mathbb{R}^d \times \mathbb{R}^d$ are such that $\|\nabla_y f(x,w)\| \leq \varepsilon$ then*

$$\|w - \text{argmax}_{z\in\mathbb{R}^d} f(x,z)\| \leq \frac{\varepsilon}{\alpha}$$

.

Proof. Let $z^\star$ be the unique global maximum point of $f(x,\cdot)$. Since $f(x,\cdot)$ is α-strongly concave, we have that

$$\alpha\|w - z^\star\| \leq \|\nabla_y f(x,w)\| \leq \varepsilon \tag{37}$$

Therefore, (37) implies that

$$\|w - \text{argmax}_{z\in\mathbb{R}^d} f(x,z)\| = \|w - z^\star\| \leq \frac{\varepsilon}{\alpha}.$$

□

Lemma A.5. *For any $x, \theta \in \mathbb{R}^d$,*

$$|\mathcal{L}_\varepsilon(x,y) - \mathcal{L}_\varepsilon(\theta,y)| \leq L\left(2\frac{\varepsilon}{\alpha} + \frac{L}{\alpha}\|\theta - x\|\right)$$

Proof. Denote by $\hat{P}_\varepsilon(x,y) \subseteq P_\varepsilon(x,y)$ the collection of endpoints of ε-greedy paths where the endpoint z of the path satisfies $\nabla_y f(x,z) = \varepsilon$. Since f is smooth, we have that $\sup_{z\in P_\varepsilon(x,y)} f(x,z) = \sup_{z\in\hat{P}_\varepsilon(x,y)} f(x,z)$ (this is true since, if the endpoint z of an ε-greedy path does not satisfy $\|\nabla_y f(x,z)\| = \varepsilon$, then the ε-greedy path can be extended to achieve a higher value of f). Thus, we

have that

$$
\begin{aligned}
|\mathcal{L}_\varepsilon(x,y) - \mathcal{L}_\varepsilon(\theta,y)| &= |\sup_{z\in P_\varepsilon(x,y)} f(x,z) - \sup_{w\in P_\varepsilon(\theta,y)} f(\theta,w)| \\
&= |\sup_{z\in \hat{P}_\varepsilon(x,y)} f(x,z) - \sup_{w\in \hat{P}_\varepsilon(\theta,y)} f(\theta,w)| \\
&\leq \sup_{z\in \hat{P}_\varepsilon(x,y), w\in \hat{P}_\varepsilon(\theta,y)} |f(x,z) - f(\theta,w)| \\
&\leq \sup_{z\in \hat{P}_\varepsilon(x,y), w\in \hat{P}_\varepsilon(\theta,y)} L \times \|w - z\| \\
&\leq \sup_{z\in \hat{P}_\varepsilon(x,y), w\in \hat{P}_\varepsilon(\theta,y)} L \times \big(\|w - \operatorname{argmax}_{\zeta\in\mathbb{R}^d} f(\theta,\zeta)\| \\
&\qquad + \|\operatorname{argmax}_{\zeta\in\mathbb{R}^d} f(\theta,\zeta) - \operatorname{argmax}_{\zeta\in\mathbb{R}^d} f(x,\zeta)\| + \|\operatorname{argmax}_{\zeta\in\mathbb{R}^d} f(x,\zeta) - z\|\big) \\
&\overset{\text{Lemmas A.3, A.4}}{\leq} L \times \left(\frac{\varepsilon}{\alpha} + \frac{L}{\alpha}\|\theta - x\| + \frac{\varepsilon}{\alpha}\right) \\
&= L \times \left(2\frac{\varepsilon}{\alpha} + \frac{L}{\alpha}\|\theta - x\|\right),
\end{aligned}
$$

where the last inequality holds by Lemma A.3, and also by Lemma A.4 because $\|\nabla_y f(x,z)\| = \varepsilon$ whenever $z \in \hat{P}_\varepsilon(x,y)$ and $\|\nabla_y f(\theta,w)\| = \varepsilon$ whenever $w \in P_\varepsilon(\theta,y)$. □

Lemma A.6. *For any $x, y \in \mathbb{R}^d$ we have*

$$|\mathcal{L}_\varepsilon(x,y) - \mathcal{L}_0(x,y)| \leq L\frac{\varepsilon^2}{2\alpha^2}$$

Proof.

$$
\begin{aligned}
|\mathcal{L}_\varepsilon(x,y) - \mathcal{L}_0(x,y)| &= |\sup_{z\in \hat{P}(x,y)} f(x,z) - f(x, \operatorname{argmax}_{z\in\mathbb{R}^d} f(x,z))| \\
&\leq \sup_{z\in \hat{P}(x,y)} |f(x,z) - f(x, \operatorname{argmax}_{z\in\mathbb{R}^d} f(x,z))| \\
&\leq \sup_{z\in \hat{P}(x,y)} \frac{L}{2}\|z - \operatorname{argmax}_{z\in\mathbb{R}^d} f(x,z)\|^2 \\
&\overset{\text{Lemmas A.4}}{\leq} \frac{L}{2}\left(\frac{\varepsilon}{\alpha}\right)^2,
\end{aligned}
$$

where the last inequality holds by Lemma A.4 because $\|\nabla_y f(x,z)\| = \varepsilon$ whenever $z \in \hat{P}_\varepsilon(x,y)$. □

Lemma A.7. *$\mathcal{L}_0(x^\star, y^\star)$ is differentiable and*

$$\|\nabla_x f(x^\star, y^\star) - \nabla_x \mathcal{L}_0(x^\star, y^\star)\| \leq L\frac{\varepsilon}{\alpha}.$$

Proof. Since $f(x,\cdot)$ is concave, we have that $\mathcal{L}_0(x^\star, y^\star) = \max_{z\in\mathbb{R}^d} f(x^\star, z)$. Moreover, by strong concavity $f(x^\star,\cdot)$ has a unique maximizer $\operatorname{argmax}_{z\in\mathbb{R}^d} f(x^\star, z)$. Thus, by Danskin's Theorem [10], we have that $\mathcal{L}_0(x^\star,\cdot)$ is differentiable and that

$$\nabla_x \mathcal{L}_0(x^\star, y^\star) = \nabla_x f(x^\star, \operatorname{argmax}_{z\in\mathbb{R}^d} f(x^\star, z)). \tag{38}$$

Therefore, since f is L-smooth,

$$\begin{aligned}\|\nabla_x f(x^\star, y^\star) - \nabla_x \mathcal{L}_0(x^\star, y^\star)\| &\overset{\text{Eq. } 38}{=} \|\nabla_x f(x^\star, y^\star) - \nabla_x f(x^\star, \operatorname{argmax}_{z\in\mathbb{R}^d} f(x^\star, z))\| \\ &\leq L \times \|y^\star - \operatorname{argmax}_{z\in\mathbb{R}^d} f(x^\star, z)\| \\ &\leq L \times \frac{\varepsilon}{\alpha},\end{aligned}$$

where the last inequality holds by Lemma A.4 since $\|\nabla_y f(x^\star, y^\star)\| \leq \varepsilon$.

□

Proof of Theorem A.2. Since $(x^\star, y^\star)$ is an $(\varepsilon, \delta, \omega, Q)$-approximate local equilibrium of f, we have that,

$$\|\nabla_y f(x^\star, y^\star)\| \leq \varepsilon,$$

and that, with probability at least $1 - \omega$,

$$\mathcal{L}_\varepsilon(x^\star - \Delta, y^\star) \geq f(x^\star, y^\star) - \delta,$$

where $\Delta \sim Q_{x^\star, y^\star}$. Thus, since by Assumption A.1 $\mathbb{E}_{\Delta\sim Q_{x,y}}[\Delta] = \mu\nabla_x f(x, y)$ and $\mathbb{E}_{\Delta\sim Q_{x,y}}[\|\Delta - \mu\nabla_x\|^2] \leq \sigma^2$, by Chebyshev's inequality, we have that, with probability at least $1 - \omega - \frac{1}{4}$,

$$\mathcal{L}_\varepsilon(x^\star - \mu\nabla_x f(x^\star, y^\star) + \nu, y^\star) \geq f(x^\star, y^\star) - \delta, \tag{39}$$

for some $\nu \in \mathbb{R}^d$ such that $\|\nu\| \leq 2\sigma$.

Since $\omega \leq \frac{1}{2}$, we have $1 - \omega - \frac{1}{4} \geq \frac{1}{4}$. Thus, (39) holds with probability at least $\frac{1}{4}$. Since (39) holds with probability at least $\frac{1}{4}$ yet contains no random variables, it must also hold with probability 1. Therefore, by plugging in Lemma A.6 to the LHS of (39) and applying Lemma A.4 together with the fact that f is L-smooth to the RHS of (39), we have that

$$\mathcal{L}_0(x^\star - \mu\nabla_x f(x^\star, y^\star) + \nu, y^\star) + L\frac{\varepsilon^2}{2\alpha^2} \geq \mathcal{L}_0(x^\star, y^\star) - L\frac{\varepsilon}{\alpha} - \delta. \tag{40}$$

Therefore, applying Lemma A.7 to (40) we get that

$$\mathcal{L}_0(x^\star - \mu\nabla_x \mathcal{L}_0(x^\star, y^\star) + v, y^\star) + L\frac{\varepsilon^2}{2\alpha^2} \geq \mathcal{L}_0(x^\star, y^\star) - L\frac{\varepsilon}{\alpha} - \delta, \tag{41}$$

for some $v \in \mathbb{R}^d$ such that $\|v\| \leq L\frac{\varepsilon}{\alpha} + 2\sigma$. Therefore, plugging in Lemma A.7 to the LHS of (41), we get that

$$\mathcal{L}_0(x^\star - \mu\nabla_x \mathcal{L}_0(x^\star, y^\star), y^\star) + L\left(2\frac{\varepsilon}{\alpha} + \mu\frac{L}{\alpha}\left(L\frac{\varepsilon}{\alpha} + 2\sigma\right)\right) + L\frac{\varepsilon^2}{2\alpha^2} \geq \mathcal{L}_0(x^\star, y^\star) - L\frac{\varepsilon}{\alpha} - \delta, \tag{42}$$

since $\|v\| \leq L\frac{\varepsilon}{\alpha} + 2\sigma$. We also have that

$$\mathcal{L}_0(x^\star - \mu\nabla_x \mathcal{L}_0(x^\star, y^\star), y^\star) = \mathcal{L}_0(x^\star, y^\star) - (\nabla_x \mathcal{L}_0(x^\star, y^\star) + u)^\top \mu\nabla_x \mathcal{L}_0(x^\star, y^\star), \tag{43}$$

for some $u \in \mathbb{R}^d$ such that $\|u\| \leq L\|\mu\nabla_x \mathcal{L}_0(x^\star, y^\star)\|$, since f is L-smooth. Therefore (43) implies that

$$\mathcal{L}_0(x^\star - \mu\nabla_x \mathcal{L}_0(x^\star, y^\star), y^\star) \leq \mathcal{L}_0(x^\star, y^\star) - (\mu - \mu^2 L)\|\nabla_x \mathcal{L}_0(x^\star, y^\star)\|^2, \tag{44}$$

Plugging (44) into (42), we get that (since $\mu = \frac{1}{2L}$ implies that $\mu - \mu^2 L > 0$),

$$\|\nabla_x \mathcal{L}_0(x^\star, y^\star)\| \leq \frac{1}{\mu - \mu^2 L}\sqrt{\delta + L\left(2\frac{\varepsilon}{\alpha} + \mu\frac{L}{\alpha}\left(L\frac{\varepsilon}{\alpha} + 2\sigma\right)\right) + L\frac{\varepsilon^2}{2\alpha^2} + L\frac{\varepsilon}{\alpha}} \tag{45}$$

But from (38) we have that

$$\nabla_x \mathcal{L}_0(x^\star, y^\star) = \nabla_x f(x^\star, \operatorname{argmax}_{z\in\mathbb{R}^d} f(x^\star, z)). \tag{46}$$

Thus,

$$\begin{aligned}\|\nabla_x \mathcal{L}_0(x^\star, y^\star) - \nabla_x f(x^\star, y^\star)\| &\overset{\text{Eq. } 46}{=} \|\nabla_x f(x^\star, \operatorname{argmax}_{z\in\mathbb{R}^d} f(x^\star, z)) - \nabla_x f(x^\star, y^\star)\| \\ &\leq L\|y^\star - \operatorname{argmax}_{z\in\mathbb{R}^d} f(x^\star, z)\| \\ &\overset{\text{Lemma A.4}}{\leq} L\frac{\varepsilon}{\alpha},\end{aligned} \tag{47}$$

where the first inequality holds since f is L-smooth, and the second inequality holds by Lemma A.4 since $\|\nabla_y f(x^\star, y^\star)\| \leq \varepsilon$.

Plugging in (47) into (45), we get

$$\|\nabla_x f(x^\star, y^\star)\| \leq \frac{1}{\mu - \mu^2}\sqrt{\delta + L\left(2\frac{\varepsilon}{\alpha} + \mu\frac{L}{\alpha}\left(L\frac{\varepsilon}{\alpha} + 2\sigma\right)\right) + L\frac{\varepsilon^2}{2\alpha^2} + L\frac{\varepsilon}{\alpha}} + L\frac{\varepsilon}{\alpha}. \tag{48}$$

Now, since $f(\cdot, y^\star)$ is α-strongly convex, by Lemma A.4 (applied to $-f$ instead of f), we have

$$\|x^\star - \operatorname{argmin}_{\theta\in\mathbb{R}^d} f(\theta, y^\star)\| \leq \frac{\|\nabla_x f(x^\star, y^\star)\|}{\alpha}. \tag{49}$$

Since f is L-smooth, (49) implies that

$$\begin{aligned}f(x^\star, y^\star) - \min_{\theta\in\mathbb{R}^d} f(\theta, y^\star) &\leq L \times \frac{1}{2}\|x^\star - \operatorname{argmin}_{\theta\in\mathbb{R}^d} f(\theta, y^\star)\|^2 \\ &\overset{\text{Eq. } 49}{\leq} \frac{L}{2} \times \left(\frac{\|\nabla_x f(x^\star, y^\star)\|}{\alpha}\right)^2.\end{aligned} \tag{50}$$

Moreover, since $f(x^\star, \cdot)$ is α-strongly concave and $\|\nabla_y f(x^\star, y^\star)\| \leq \varepsilon$ we have by Lemma A.4 that

$$\|y^\star - \operatorname{argmax}_{z\in\mathbb{R}^d} f(x^\star, z)\| \leq \frac{\varepsilon}{\alpha}. \tag{51}$$

Since f is L-smooth, (51) implies that

$$\begin{aligned}\max_{z\in\mathbb{R}^d} f(x^\star, z) - f(x^\star, y^\star) &\leq L \times \frac{1}{2}\|y^\star - \operatorname{argmax}_{z\in\mathbb{R}^d} f(x^\star, z)\|^2 \\ &\overset{\text{Eq. } 51}{\leq} \frac{L}{2} \times \left(\frac{\varepsilon}{\alpha}\right)^2.\end{aligned} \tag{52}$$

Thus, adding (50) to (52), and plugging in (48), we get that

$$\max_{z\in\mathbb{R}^d} f(x^\star,z) - \min_{\theta\in\mathbb{R}^d} f(\theta,y^\star) \le \frac{L}{2}\times\left(\frac{\|\nabla_x f(x^\star,y^\star)\|}{\alpha}\right)^2 + \frac{L}{2}\times\left(\frac{\varepsilon}{\alpha}\right)^2$$
$$\overset{\text{Eq. 48}}{\le} \frac{L\varepsilon^2}{2\alpha^2} + \frac{L}{2\alpha^2}\left(\frac{1}{\mu-\mu^2}\sqrt{\delta + L\left(2\frac{\varepsilon}{\alpha}+\mu\frac{L}{\alpha}\left(L\frac{\varepsilon}{\alpha}+2\sigma\right)\right) + L\frac{\varepsilon^2}{2\alpha^2}+L\frac{\varepsilon}{\alpha}} + L\frac{\varepsilon}{\alpha}\right)^2.$$

Since $\mu = \frac{1}{2L}$, we get that

$$\begin{aligned}
&\max_{z\in\mathbb{R}^d} f(x^\star,z) - \min_{\theta\in\mathbb{R}^d} f(\theta,y^\star) \\
&\qquad\le \frac{L\varepsilon^2}{2\alpha^2} + \frac{L}{2\alpha^2}\left(\frac{1}{\mu-\mu^2}\sqrt{\delta + L\left(2\frac{\varepsilon}{\alpha}+\mu\frac{L}{\alpha}\left(L\frac{\varepsilon}{\alpha}+2\sigma\right)\right) + L\frac{\varepsilon^2}{2\alpha^2}+L\frac{\varepsilon}{\alpha}} + L\frac{\varepsilon}{\alpha}\right)^2 \\
&\qquad= \frac{L\varepsilon^2}{2\alpha^2} + \frac{L^3}{\alpha^2}\left(\sqrt{\delta + L\left(2\frac{\varepsilon}{\alpha}+\frac{1}{\alpha}\left(L\frac{\varepsilon}{\alpha}+2\sigma\right)\right) + L\frac{\varepsilon^2}{2\alpha^2}+L\frac{\varepsilon}{\alpha}} + L\frac{\varepsilon}{\alpha}\right)^2.
\end{aligned}$$

□

A.1 Runtime in Strongly Convex-Strongly Concave Setting

Suppose that $f(x,y)$ is L-smooth in (x,y), α-strongly convex in x and α-strongly concave in y.

In this section we assume that hyper-parameter τ_1 in Algorithm 1 is set to $\tau_1 = \infty$, and that Algorithm 2 takes as input exact gradients for $\nabla_y f$ (i.e., the "stochastic" gradients have variance set to 0).

Lemma A.8. *The function* $\max_{z\in\mathbb{R}^d} f(\cdot,z)$ *is* α*-strongly convex.*

Proof. Define the "global max" function $\psi(x) := \max_{z\in\mathbb{R}^d} f(x,z)$ for all $x\in\mathcal{X}$. We start by showing that the function $\psi(x)$ is α-strongly convex. Indeed, for any $x_1,x_2\in\mathcal{X}$ and any $\lambda\in[0,1]$ we have

$$\begin{aligned}
\lambda\psi(\lambda x_1+(1-\lambda)x_2) &= \max_{y\in\mathbb{R}^d} f(\lambda x_1+(1-\lambda)x_2,y) \\
&\le \max_{y\in\mathbb{R}^d}\left[\lambda f(x_1,y)+(1-\lambda)f(x_2,y) - \frac{1}{2}\alpha\lambda(1-\lambda)\|x_1-x_2\|^2\right] \\
&\le \lambda\left[\max_{y\in\mathbb{R}^d} f(x_1,y)\right] + (1-\lambda)[\max_{y\in\mathbb{R}^d} f(x_2,y)\Big] - \frac{1}{2}\alpha\lambda(1-\lambda)\|x_1-x_2\|^2 \\
&= \lambda\psi(x_1)+(1-\lambda)\psi(x_2) - \frac{1}{2}\alpha\lambda(1-\lambda)\|x_1-x_2\|^2,
\end{aligned}$$

where the first inequality holds by the α-strong convexity of $f(\cdot,y)$. Thus ψ is α-strongly convex. □

Lemma A.9. *Denote by* $\mathsf{y}_{i,j}$ *the point* y_j *in Algorithm 2 when it is called at the* i*'th iteration of Algorithm 1. Let* $(x^\dagger,y^\dagger)$ *be the global min-max point of* f*, and define* $D := \|(x_0,y_0)-(x^\dagger,y^\dagger)\|$ *and*

$$\mathfrak{D} := 2\max\left(D+\frac{LD}{\alpha}+\frac{\varepsilon}{\alpha}, \sqrt{\frac{LD+\frac{L^2D}{\alpha}+L\frac{\varepsilon^2}{\alpha^2}}{\alpha}}, \frac{L\sqrt{D}}{\alpha\sqrt{\alpha}}, \frac{\varepsilon\sqrt{L}}{\alpha\sqrt{\alpha}}\right).$$

Then, as long as $\eta \leq \frac{1}{L}$, at every iteration i of Algorithm 1 and at every iteration j of its subroutine Algorithm 2, we have that $\|(x_i, y_i) - (x^\dagger, y^\dagger)\| \leq \mathfrak{D}$ and $\|(x_i, \mathsf{y}_{i,j}) - (x^\dagger, y^\dagger)\| \leq \mathfrak{D}$.

Proof. **Bounding the distance $\|(x_1, y_1) - (x^\dagger, y^\dagger)\|$.**

Since at the global min-max point $(x^\dagger, y^\dagger)$ we have $\nabla_y f(x^\dagger, y^\dagger) = 0$, and since f is L-Lipschitz, we have that

$$\|\nabla_y f(x_0, y_0)\| \leq L\|(x_0, y_0) - (x^\dagger, y^\dagger)\| \leq LD.$$

Thus, since $f(x_0, \cdot)$ is α-strongly concave, we have by Lemma A.4 that

$$\|y_0 - \text{argmax}_{z \in \mathbb{R}^d} f(x_0, z)\| \leq \frac{LD}{\alpha}. \tag{53}$$

And since (by definition) $x_0 = x_1$, and $\nabla_y f(x_0, y_1) = \nabla_y f(x_1, y_1) \leq \varepsilon$, we have that (again by Lemma A.4),

$$\|y_1 - \text{argmax}_{z \in \mathbb{R}^d} f(x_0, z)\| \leq \frac{\varepsilon}{\alpha}. \tag{54}$$

Thus, combining (53) and (54), we have that (since $x_0 = x_1$),

$$\begin{aligned}
\|(x_1, y_1) - (x^\dagger, y^\dagger)\| &\leq \|(x_0, y_0) - (x^\dagger, y^\dagger)\| + \|(x_0, y_0) - (x_0, y_1)\| \\
&\leq D + \|y_0 - y_1\| \\
&\leq D + \|y_1 - \text{argmax}_{z \in \mathbb{R}^d} f(x_0, z)\| + \|y_0 - \text{argmax}_{z \in \mathbb{R}^d} f(x_0, z)\| \\
&\leq D + \frac{LD}{\alpha} + \frac{\varepsilon}{\alpha} \\
&\leq \mathfrak{D}.
\end{aligned}$$

Bounding the distance $\|x_i - x^\dagger\|$.

At each iteration $i > 1$ of Algorithm 1 we have that $\|\nabla_y f(x_i, y_i)\| \leq \varepsilon$. Thus, by Lemma A.4 we have that

$$\|y_i - \text{argmax}_{z \in \mathbb{R}^d} f(x_i, z)\| \leq \frac{\varepsilon}{\alpha}. \tag{55}$$

Thus, since f is L-smooth,

$$\begin{aligned}
\max_{z \in \mathbb{R}^d} f(x_i, z) - f(x_i, y_i) &\leq L \times \|y_i - \text{argmax}_{z \in \mathbb{R}^d} f(x_i, z)\|^2 \\
&\overset{\text{Eq. } 55}{\leq} L\frac{\varepsilon^2}{\alpha^2}.
\end{aligned} \tag{56}$$

But $f(x_{i+1}, y_{i+1}) \leq f(x_i, y_i)$ at each iteration i (since, if the proposed update to x_i does not lead to a decrease in the value of f we have that the proposed update to x_i would be rejected and $x_i = x_{i+1}$ and $y_i = y_{i+1}$). Therefore,

$$f(x_i, y_i) \leq f(x_1, y_1) \qquad \forall i \geq 1. \tag{57}$$

Therefore, (56) and (57) imply that

$$\begin{aligned}
\max_{z\in\mathbb{R}^d} f(x_i, z) &\leq f(x_i, y_i) + L\frac{\varepsilon^2}{\alpha^2} \\
&\overset{\text{Eq. 56}}{\leq} f(x_1, y_1) + L\frac{\varepsilon^2}{\alpha^2} \\
&\overset{\text{Eq. 57}}{\leq} \max_{z\in\mathbb{R}^d} f(x_1, z) + L\frac{\varepsilon^2}{\alpha^2} \\
&= f(x_0, \operatorname{argmax}_{z\in\mathbb{R}^d} f(x_0, z)) + L\frac{\varepsilon^2}{\alpha^2} \\
&\overset{\text{Eq. 53}}{\leq} f(x_0, y_0) + L \times \frac{LD}{\alpha} + L\frac{\varepsilon^2}{\alpha^2}, \qquad (58)
\end{aligned}$$

since $(x_0 = x_1)$, and where the last inequality holds by (53) since f is L-smooth. But since $\nabla_x f(x^\dagger, y^\dagger) = \nabla_y f(x^\dagger, y^\dagger) = 0$, and f is L-smooth,

$$f(x^\dagger, y^\dagger) - f(x_0, y_0) \leq L\|(x^\dagger, y^\dagger) - (x_0, y_0)\| \leq LD. \qquad (59)$$

Thus, plugging in (59) into (58), we get

$$\begin{aligned}
\psi(x_i) - \min_{\theta\in\mathbb{R}^d} \psi(\theta) &= \max_{z\in\mathbb{R}^d} f(x_i, z) - \min_{\theta\in\mathbb{R}^d} \max_{z\in\mathbb{R}^d} f(\theta, z) \\
&= \max_{z\in\mathbb{R}^d} f(x_i, z) - f(x^\dagger, y^\dagger) \\
&\overset{\text{Eq. 53}}{\leq} LD + \frac{L^2 D}{\alpha} + L\frac{\varepsilon^2}{\alpha^2}. \qquad (60)
\end{aligned}$$

But we have shown that ψ is α-strongly convex. Therefore, (60) implies that

$$\begin{aligned}
\|x_i - x^\dagger\| &= \|x_i - \operatorname{argmin}_{\theta\in\mathbb{R}^d}\psi(\theta)\| \\
&\leq \sqrt{\frac{LD + \frac{L^2 D}{\alpha} + L\frac{\varepsilon^2}{\alpha^2}}{\alpha}} \\
&\leq \mathfrak{D}. \qquad (61)
\end{aligned}$$

Bounding the distance $\|\mathsf{y}_{i,j} - y^\dagger\|$.
Now, since $\eta \leq \frac{1}{L}$, we have that $f(x_i, \mathsf{y}_{i,j})$ is nondecreasing at each iteration j of Algorithm 2,

$$f(x_i, \mathsf{y}_{i,j+1}) \geq f(x_i, \mathsf{y}_{i,j}) \qquad \forall i \geq 0, j \geq 1. \qquad (62)$$

First we consider the case when $i = 0$. Since $\mathsf{y}_{1,0} = y_0$, by (53) we have that

$$\|\mathsf{y}_{1,0} - \operatorname{argmax}_{z\in\mathbb{R}^d} f(x_0, z)\| \leq \frac{LD}{\alpha}. \qquad (63)$$

Thus, since $f(x_0, \cdot)$ is L-smooth, (63) implies that

$$\max_{z\in\mathbb{R}^d} f(x_0, z) - f(x_0, \mathsf{y}_{1,0}) \leq L\left(\frac{\sqrt{LD}}{\alpha}\right)^2.$$

Thus, by (62) we have that

$$\max_{z\in\mathbb{R}^d} f(x_0,z) - f(x_0,\mathsf{y}_{0,j}) \le L\left(\frac{\sqrt{LD}}{\alpha}\right)^2 \qquad \forall j \ge 1. \tag{64}$$

Thus, since $\nabla_y f(x_0, \operatorname{argmax}_{z\in\mathbb{R}^d} f(x_0,z)) = 0$, and since $f(x_0,\cdot)$ is α-strongly concave, we have that

$$\|\mathsf{y}_{1,j} - \operatorname{argmax}_{z\in\mathbb{R}^d} f(x_0,z)\| \le \sqrt{\frac{L}{\alpha}\left(\frac{\sqrt{LD}}{\alpha}\right)^2} = \frac{L\sqrt{D}}{\alpha\sqrt{\alpha}} \le \mathfrak{D} \qquad \forall j \ge 1. \tag{65}$$

Next, we consider the case when $i \ge 1$. At each $i \ge 1$, we have that $\|\nabla_y f(x_i,y_i)\| = \|\nabla_y f(x_i,\mathsf{y}_{i,0})\| \le \varepsilon$. Therefore, since $f(x_i,\cdot)$ is α-strongly concave, we have by Lemma A.4 that

$$\|\mathsf{y}_{i,0} - \operatorname{argmax}_{z\in\mathbb{R}^d} f(x,z)\| \le \frac{\varepsilon}{\alpha}. \tag{66}$$

Thus, since $f(x_i,\cdot)$ is L-smooth, (66) implies that

$$\max_{z\in\mathbb{R}^d} f(x_i,z) - f(x_i,\mathsf{y}_{i,0}) \le L\left(\frac{\varepsilon}{\alpha}\right)^2.$$

Thus by (62) we have that

$$\max_{z\in\mathbb{R}^d} f(x_i,z) - f(x_i,\mathsf{y}_{i,j}) \le L\left(\frac{\varepsilon}{\alpha}\right)^2 \qquad \forall j \ge 0.$$

Thus, since $\nabla_y f(x_0, \operatorname{argmax}_{z\in\mathbb{R}^d} f(x_0,z)) = 0$, and since $f(x_0,\cdot)$ is α-strongly concave, we have that

$$\|\mathsf{y}_{i,j} - \operatorname{argmax}_{z\in\mathbb{R}^d} f(x_i,z)\| \le \sqrt{\frac{L}{\alpha}\left(\frac{\varepsilon}{\alpha}\right)^2} = \frac{\varepsilon\sqrt{L}}{\alpha\sqrt{\alpha}} \le \mathfrak{D} \qquad \forall j \ge 1. \tag{67}$$

Therefore, from (61), (64), and (67), and since $y_i = \mathsf{y}_{i,0}$ for every i, we have that $\|(x_i,y_i) - (x^\dagger,y^\dagger)\| \le \mathfrak{D}$ and $\|(x_i,\mathsf{y}_{i,j}) - (x^\dagger,y^\dagger)\| \le \mathfrak{D}$ for every $i \ge 0$ and every $j \ge 0$

□

Corollary A.10. *Suppose that $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is such that $f(\cdot,y)$ is α-strongly convex for every $y \in \mathbb{R}^d$ and $f(x,\cdot)$ is α-strongly concave for every $x \in \mathbb{R}^d$, and that f has L-Lipschitz gradients for some $L \ge \alpha > 0$. And suppose that the proposal distribution $Q_{x,y}$ of Algorithm 1 are the deterministic gradients $\Delta = -\frac{1}{2L}\nabla_x f(x,y)$ for $\Delta \sim Q_{x,y}$, and that Algorithm 2 takes as input deterministic gradients $\nabla_y f$. Then, given any $\varepsilon' > 0$ and any initial point $(x_0,y_0) \in \mathbb{R}^d \times \mathbb{R}^d$, Algorithm 1, with appropriate parameters, outputs a point $(x^\star,y^\star)$ which is an approximate global min-max point of f with duality gap*

$$\max_{y\in\mathbb{R}^d} f(x^\star,y) - \min_{x\in\mathbb{R}^d} f(x,y^\star) \le \varepsilon'$$

in $\operatorname{poly}\left(L, \frac{1}{\alpha}, \frac{1}{\varepsilon'}, D\right)$ *gradient and function evaluations, where $D = \|(x_0,y_0) - (x^\dagger,y^\dagger)\|$ is the distance from the initial point to the (exact) global min-max point $(x^\dagger,y^\dagger)$ of f.*

Proof. Set the parameters

$$\varepsilon = \frac{1}{10}\min\left(\frac{\alpha}{\sqrt{L}}\sqrt{\varepsilon'},\ \frac{\alpha^4}{L^5}\varepsilon'\right), \delta = \frac{1}{10}\frac{\alpha^2}{L^3}\varepsilon',\ \text{and } \omega = \frac{1}{4}.$$

Define

$$\mathfrak{D} := 2\max\left(D + \frac{LD}{\alpha} + \frac{\varepsilon}{\alpha},\ \sqrt{\frac{LD + \frac{L^2D}{\alpha} + L\frac{\varepsilon^2}{\alpha^2}}{\alpha}},\ \frac{L\sqrt{D}}{\alpha\sqrt{\alpha}},\ \frac{\varepsilon\sqrt{L}}{\alpha\sqrt{\alpha}}\right).$$

Define $b := 4L\mathfrak{D}^2$, and $L_1 := 2L\mathfrak{D}$. Set hyperparameters $\tau_1 = \infty$ (so that the rejection probability in line 10 of Algorithm 1 is 1). Set the remaining hyperparameters as in Items 1-8 in Section 6 with the parameter "L" in Items 1-8 replaced by $\min(L, \frac{L_1^2}{2b})$. Since $\eta \leq \frac{1}{10L}$, by Lemma A.9 we have that every step i of Algorithm 1 and every step j of its subroutine Algorithm 2 satisfy

$$\|(x_i, y_i) - (x^\dagger, y^\dagger)\| \leq \mathfrak{D} \text{ and } \|(x_i, \mathsf{y}_{i,j}) - (x^\dagger, y^\dagger)\| \leq \mathfrak{D}$$

for all $i \geq 0$ and all $j \geq 0$. Thus, Algorithm 1 and its subroutine Algorithm 2 remain inside the ball $B((x^\dagger, y^\dagger), \mathfrak{D})$ of radius $\mathfrak{D}$ with center at the global min-max point $(x^\dagger, y^\dagger)$ of f. Since $(x^\dagger, y^\dagger)$ is the global min-max point of f, we have that

$$\nabla_x f(x^\dagger, y^\dagger) = \nabla_y f(x^\dagger, y^\dagger) = 0.$$

Without loss of generality we may assume that $f(x^\dagger, y^\dagger) = 0$ (we can assume this since each step of the algorithm remains the same if we add a constant to f). Thus, since f is L-smooth on all of $\mathbb{R}^d \times \mathbb{R}^d$, we have that

$$|f(x,y)| \leq L \times 4\mathfrak{D}^2 \qquad \forall (x,y) \in B((x^\dagger, y^\dagger), 2\mathfrak{D}),\ \text{and}$$

$$\|(\nabla_x f(x,y), \nabla_y f(x,y))\| \leq L \times 2\mathfrak{D} \qquad \forall (x,y) \in B((x^\dagger, y^\dagger), 2\mathfrak{D}).$$

Since f is b-bounded with L_1-Lipschitz gradient on the ball $B((x^\dagger, y^\dagger), 2\mathfrak{D})$, and since every step of the algorithm remains inside the ball $B((x^\dagger, y^\dagger), \mathfrak{D}) \subseteq B((x^\dagger, y^\dagger), 2\mathfrak{D})$, each step of the proof of Theorem 3.3 holds if we replace the parameter "L" in that proof with $\min\left(L, \frac{L_1^2}{2b}\right)$ (since the parameter "L" in the proof of Theorem 3.3 is required to be $\leq \frac{L_1^2}{4b}$, and setting that parameter "L" to be $\min\left(L, \frac{L_1^2}{2b}\right)$ ensures that this assumption holds).

Therefore, the conclusion of Theorem 3.3 must also hold and we have that Algorithm 1 returns a point $(x^\star, y^\star) \in \mathbb{R}^d \times \mathbb{R}^d$ such that, for some $\varepsilon^\star \in \left[\frac{1}{2}\varepsilon, \varepsilon\right]$, $(x^\star, y^\star)$ is an $(\varepsilon^\star, \delta, \omega, Q)$-equilibrium. The number of gradient and function evaluations required by the algorithm is

$$\text{poly}\left(b, \min\left(L, \frac{L_1^2}{2b}\right), \frac{1}{\varepsilon}, \frac{1}{\delta}, \frac{1}{4}\right)$$

and does not depend on the dimension d. Note that, since we assume the gradients and proposal distribution are deterministic, each step of the algorithm is also deterministic, and the conclusion must hold with probability 1. But

$$\frac{1}{\varepsilon}, b, \frac{1}{\delta} = \text{poly}\left(L, \frac{1}{\alpha}, \frac{1}{\varepsilon}, D\right) \text{ and } \min\left(L, \frac{L_1^2}{2b}\right) = \text{poly}\left(L, \frac{1}{\alpha}, \frac{1}{\varepsilon}, D\right).$$

Therefore, the number of gradient and function evaluations is also poly$(L, \frac{1}{\alpha}, \frac{1}{\varepsilon}, D)$. We have now shown that Algorithm 1 returns a point $(x^\star, y^\star)$ which is an $(\varepsilon^\star, \delta, \omega, Q)$-equilibrium for f where $\varepsilon^\star \in \left[\frac{1}{2}\varepsilon, \varepsilon\right]$ (and in particular, $\varepsilon^\star, \delta = \text{poly}\left(\varepsilon', \alpha, \frac{1}{L}\right)$). Therefore, by Theorem A.2, we have that since $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is α-strongly convex in x and α-strongly concave in y, with L-Lipschitz gradient in both variables, the point $(x^\star, y^\star)$, which is an $(\varepsilon^\star, \delta, \omega, Q)$-equilibrium, also satisfies the duality gap

$$\begin{aligned}
&\max_{y \in \mathbb{R}^d} f(x^\star, y) - \min_{x \in \mathbb{R}^d} f(x, y^\star) \\
&\qquad \leq \frac{L(\varepsilon^\star)^2}{2\alpha^2} + \frac{L^3}{\alpha^2}\left(\sqrt{\delta + L\left(2\frac{\varepsilon^\star}{\alpha} + \frac{1}{\alpha}\left(L\frac{\varepsilon^\star}{\alpha} + 2\sigma\right)\right) + L\frac{(\varepsilon^\star)^2}{2\alpha^2} + L\frac{\varepsilon^\star}{\alpha}} + L\frac{\varepsilon^\star}{\alpha}\right)^2 \\
&\qquad \leq \varepsilon'.
\end{aligned}$$

□

B Examples of Functions where Global Min-Max Satisfies Definition 3.2 but not Other Local Equilibrium Notions

In this section, we expand upon the examples mentioned in Section 3. In particular, we provides example functions for which there exists min-max points that satisfy Definition 3.2 but which do not satisfy other common notions of local equilibrium.

Functions for which global min-max points are not first-order stationary points. Consider the function

$$f(x, y) = \sin(x) \times \sin(y) - \sum_{m,n \in \mathbb{Z}} \text{Bump}(x + m\pi, y + n\pi),$$

where

$$\text{Bump}(x, y) = e^{-1/(1-100(x^2+y^2))} \text{ for } x^2 + y^2 < \frac{1}{100} \text{ and Bump}(x, y) = 0 \text{ everywhere else.}$$

This function has a global min-max point at $(x, y) = (0, 1)$ and this point also satisfies Definition 3.2 (and f also has such a point at all points along the line $x = 0$ except for the intervals $(-\frac{1}{10}+n\pi, \frac{1}{10}+n\pi)$ for intergers n), and yet

$$\nabla_x f(0, 1) = \cos(0) \times \sin(1) = 0.84$$

meaning that $(x, y) = (0, 1)$ is not a first-order stationary point for x. In fact, every global min-max point of this function is not a first-order stationary point in x.

Functions for which global min-max points are not second-order equilibrium points. For functions

$$f(x, y) = \sin(x + y) \text{ and}$$
$$f(x, y) = 10^3 \cdot \sum_{k \in \mathbb{Z}} e^{-(x+y+2+9k)^2} + 2e^{-(x+y+2+9k)^2} - e^{-(x+6k)^2},$$

there are no ε-approximate local min/max points for $\varepsilon < \frac{1}{2}$, and yet, an equilibrium point from Definition 3.2 is guaranteed to exist for such functions. Note that these functions are indeed smooth and bounded.

C Comparison of Local Equilibrium Point and Local Min-Max Point

Lemma C.1. *Suppose that $(x^\star, y^\star)$ is such that $y^\star$ is a local maximum point of $f(x^\star, \cdot)$ and $x^\star$ is a local minimum point of $f(\cdot, y^\star)$. Then $(x^\star, y^\star)$ is also a local equilibrium of f.*

Proof. Fix any $\varepsilon \geq 0$ (the proof of this Lemma requires only $\varepsilon = 0$, but we state the proof for any $\varepsilon \geq 0$ since this will allow us to prove Corollary C.2). Since $y^\star$ is a local maximum of $f(x^\star, \cdot)$, there is only one ε-greedy path with initial point $y^\star$, namely, the path $\{y^\star\}$ consisting of the single point $y^\star$ (since f must increase at rate at least ε at every point on an ε-greedy path). Thus,

$$P_\varepsilon(x^\star, y^\star) = \{y^\star\} \tag{68}$$

Hence, (68) implies that

$$y^\star \in \text{argmax}_{y \in P_\varepsilon(x^\star, y^\star)} f(x^\star, y) \tag{69}$$

which proves Equation (5). Next, we will show that Equation (4) holds. Since $x^\star$ is a local minimum point of $f(\cdot, y^\star)$, there exists $\nu > 0$ such that

$$f(z, y^\star) \geq f(x^\star, y^\star) \qquad \forall z \in B(x^\star, \nu) \tag{70}$$

Since $y^\star \in P_\varepsilon(x, y^\star)$ for all $x \in \mathcal{X}$, we have that

$$\max_{y \in P_\varepsilon(x, y^\star)} f(x, y) \geq f(x, y^\star) \qquad \forall x \in \mathcal{X}, \tag{71}$$

and hence that

$$\begin{aligned} \min_{x \in B(x^\star, \nu) \cap \mathcal{X}} \max_{y \in P_\varepsilon(x, y^\star)} f(x, y) &\overset{\text{Eq. 71}}{\geq} \min_{x \in B(x^\star, \nu)} f(x, y^\star) \\ &\overset{\text{Eq. 70}}{=} f(x^\star, y^\star) \\ &\overset{\text{Eq. 69}}{=} \max_{y \in P_\varepsilon(x^\star, y^\star)} f(x^\star, y), \end{aligned}$$

which proves Equation (4).

□

Corollary C.2. *Suppose that $(x^\star, y^\star)$ is such that $y^\star$ is a local maximum point of $f(x^\star, \cdot)$ and $x^\star$ is a local minimum point of $f(\cdot, y^\star)$. Then there exists $\nu > 0$ such that, for any $\varepsilon, \delta \geq 0$, and any proposal distribution Q with support on $\mathcal{X}$ which satisfies*

$$\Pr_{\Delta \sim Q_{x^\star, y^\star}} (\|\Delta\| \geq \nu) < \omega, \tag{72}$$

for some $\omega > 0$, $(x^\star, y^\star)$ is also an approximate local equilibrium of f for parameters $(\varepsilon, \delta, \omega)$ and proposal distribution Q.

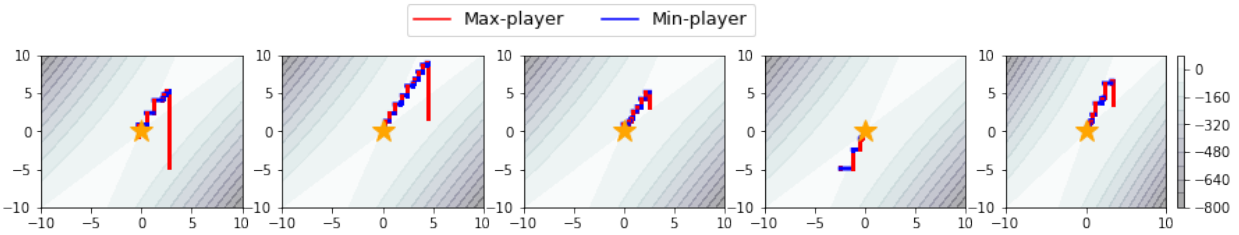

Figure 6: Different runs of our algorithm over function F_1 for random starting points.

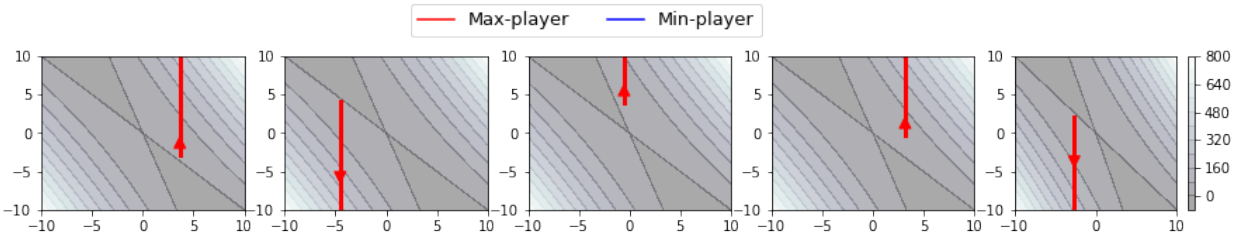

Figure 7: Different runs of our algorithm over function F_2 for random starting points.

We note that many distributions satisfy (72), for instance the distribution $Q_{x,y} \sim N(0, \sigma^2 I_d)$ for $\sigma = O(\nu \log^{-1}(\frac{1}{\omega}))$.

Proof. By Inequality (72) in the proof of Lemma C.1, there exists $\nu > 0$ such that

$$\min_{x \in B(x^\star, \nu) \cap \mathcal{X}} \max_{y \in P_\varepsilon(x, y^\star)} f(x, y) \geq \max_{y \in P_\varepsilon(x^\star, y^\star)} f(x^\star, y), \tag{73}$$

Thus, for any proposal distribution Q which satisfies Inequality (72), Inequality (73) implies that, for any $\delta \geq 0$,

$$\begin{aligned}
\Pr_{\Delta \sim Q_{x^\star, y^\star}} \left[\max_{y \in P_\varepsilon(x^\star + \Delta, y^*)} f(x^\star + \Delta, y) < \max_{y \in P_\varepsilon(x^\star, y^*)} f(x^\star, y) - \delta \right] & \\
\overset{\text{Eq. 73}}{\leq} \Pr_{\Delta \sim Q_{x^\star, y^\star}} [x^\star + \Delta \notin B(x^\star, \nu) \cap \mathcal{X}] & \\
= \Pr_{\Delta \sim Q_{x^\star, y^\star}} (\|\Delta\| \geq \nu) & \\
\overset{\text{Eq. 72}}{<} \omega, &
\end{aligned}$$

This proves Inequality (6). Inequality (7) follows directly from Inequality (69) in the proof of Lemma C.1.

□

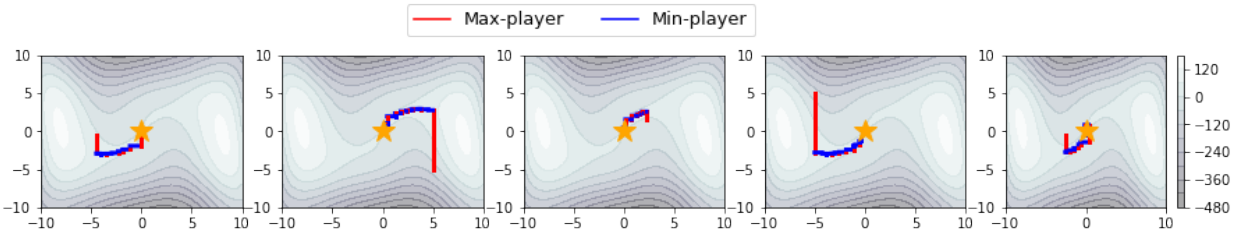

Figure 8: Different runs of our algorithm over function F_3 for random starting points.

D Additional Empirical Details and Results for Test Functions and Gaussian Mixture Dataset

D.1 Simulation Setup for Low-dimensional Test Functions

In this section we describe the setup for the simulations on the low-dimensional test functions presented in Figures 1 and 2. For our algorithm, we use a learning rate of $\eta = 0.05$ for the max-player, and a proposal distribution of $Q_{x,y} \sim N(0, 0.25)$ for the min-player. For GDA and OMD we use a learning rate of 0.05 for both the min-player and the max-player. When generating Figures 1 and 2 we used the initial point $(x_0, y_0) = (5.5, 5.5)$ for all three algorithms.

D.2 Additional Simulation Results for Low-dimensional Test Functions

We also run our algorithm for toy functions F_1, F_2, F_3 (defined in Section 5) on random initial points. The results are present in Figures 6, 7, 8 for functions F_1, F_2, F_3, respectively. For all starting points, our algorithm converges to global min-max point $(0, 0)$ for functions F_1, F_3, and diverges to ∞ for function F_2.

D.3 Simulation Setup for Gaussian Mixture Dataset

In this section we discuss the neural network architectures, choice of hyperparameters, and hardware used for the Gaussian mixture dataset

Hyperparameters for Gaussian mixture simulations. For the simulations on Gaussian mixture data, we have used the code provided by the authors of [40] (`github.com/poolio/unrolled_gan`), which uses a batch size 512, Adam learning rates of 10^{-3} for the generator and 10^{-4} for the discriminator, and Adam parameter $\beta_1 = 0.5$ for both the generator and discriminator.[3] We use the same neural networks that were used in the code from [40]: The generator uses a fully connected neural network with 2 hidden layers of size 128 and RELU activation, followed by a linear projection to two dimensions. The discriminator uses a fully connected neural network with 2 hidden layers of size 128 and RELU activation, followed by a linear projection to 1 dimension (which is fed as input

[3]Note that the authors also mention using slightly different ADAM parameters and neural network architecture in their paper than in their code; we have used the Adam parameters and neural network architecture provided in their code.

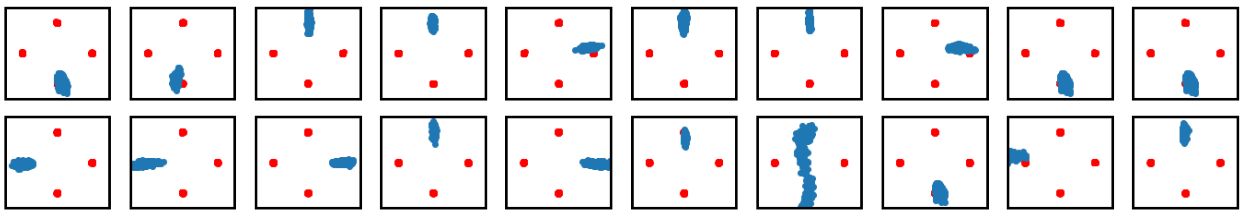

Figure 9: The generated points at the 1500'th iteration for all runs of GDA with $k = 1$ discriminator steps.

to the cross entropy loss function). As in the paper [40], we initialize all the neural network weights to be orthogonal with scaling 0.8.

For OMD, we once again use Wasserstein loss and clip parameter 0.01 (`github.com/vsyrgkanis/optimistic_GAN_training/`).

Setting hyperparameters. In our simulations, our goal was to be able to use the smallest number of discriminator or unrolled steps while still learning the distribution in a short amount of time, and we therefore decided to compare all algorithms using the same hyperparameter k. To choose this single value of k, we started by running each algorithm with $k = 1$ and increased the number of discriminator steps until one of the algorithms was able to learn the distribution consistently in the first 1500 iterations.

The experiments were performed on four 3.0 GHz Intel Scalable CPU Processors, provided by AWS.

D.4 Additional Simulation Results for Gaussian Mixture Dataset

In this section we show the results of all the runs of the simulation mentioned in Figure 3, where all the algorithms were trained on a 4-Gaussian mixture dataset for 1500 iterations. For each run, we plot points from the generated distribution at iteration 1,500. Figure 9 gives the results for GDA with $k = 1$ discriminator step. Figure 10 gives the results for GDA with $k = 6$ discriminator steps. Figure 11 gives the results for the Unrolled GANs algorithm. Figure 12 gives the results for the OMD algorithm. Figure 13 gives the results for our algorithm.

GDA with 6 discriminator steps

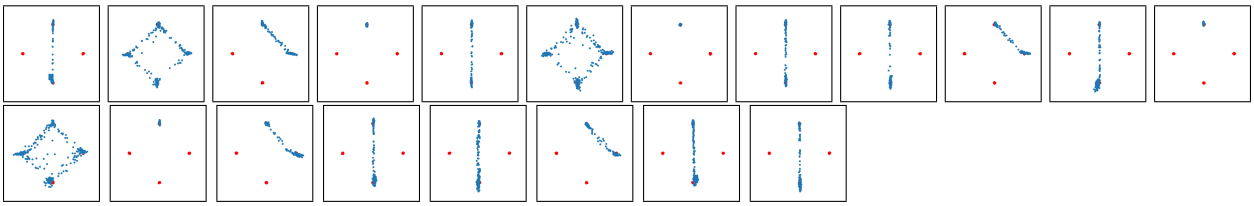

Figure 10: The generated points at the 1500'th iteration for all runs of the GDA algorithm, with $k = 6$ discriminator steps, for the simulation mentioned in Figure 3. At the 1500'th iteration, GDA had learned two modes 65% of the runs, one mode 20% of the runs, and four modes 15 % of the runs.

Unrolled GANs with 6 unrolling steps

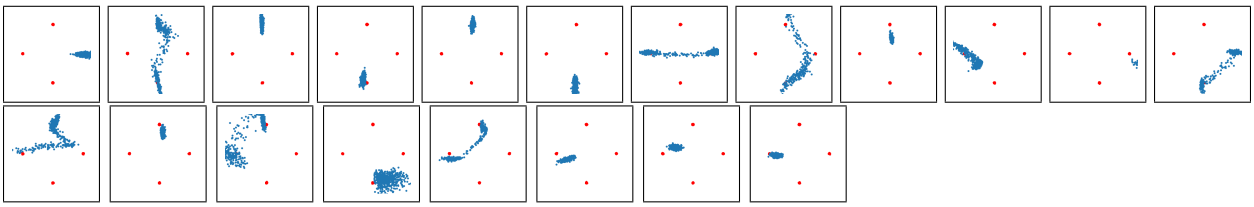

Figure 11: The generated points at the 1500'th iteration for all runs of the Unrolled GAN algorithm for the example in Figure 3, with $k = 6$ unrolling steps.

OMD

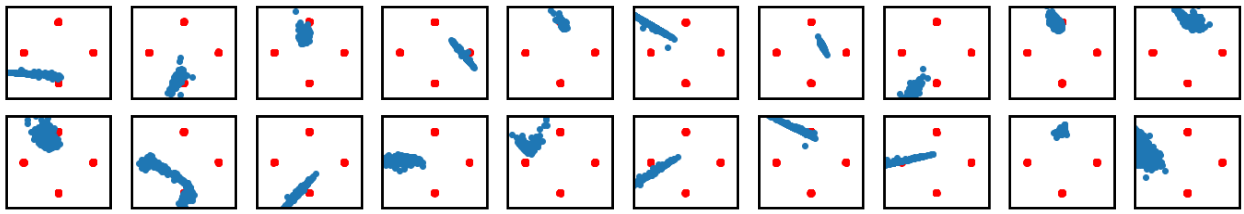

Figure 12: The generated points at the 1500'th iteration for all runs of OMD algorithm.

Our algorithm

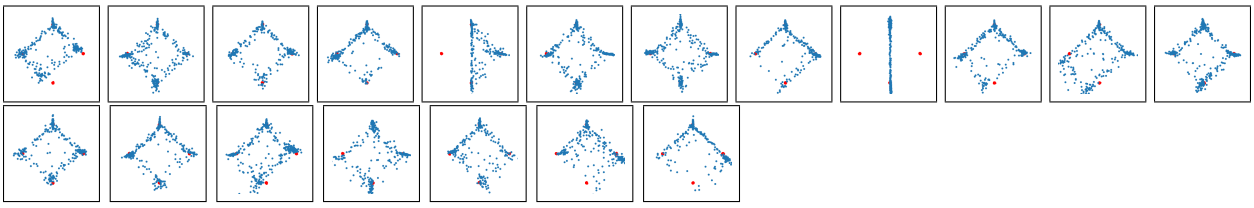

Figure 13: The generated points at the 1500'th iteration for all runs of our algorithm, for the simulation mentioned in Figure 3. Our algorithm used $k = 6$ discriminator steps and an acceptance rate hyperparameter of $\frac{1}{\tau} = \frac{1}{4}$. By the 1500'th iteration, our algorithm seems to have learned all four modes 70% of the runs, three modes 15% of the runs, and two modes 15% of the runs.

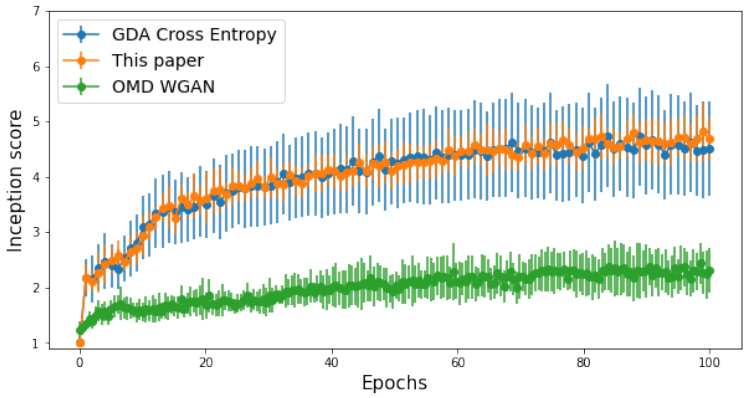

Figure 14: Inception score average (and standard deviation in errorbars) of all methods across iterations. Note that mean inception score of our algorithm is higher than the mean inception score of OMD, while the standard deviation of inception score of our algorithm is lower than the standard deviation of inception score of GDA.

Figure 15: GAN trained using our algorithm (with $k = 1$ discriminator steps and acceptance rate $e^{-\frac{1}{\tau}} = \frac{1}{2}$). We repeated this simulation multiple times; here we display images generated from some of the resulting generators for our algorithm.

Table 3: CIFAR-10 dataset: The mean (and standard error) of Inception Scores of models from different training algorithms. Note that, GDA and our algorithm return generators with similar mean performance; however, the standard error of the Inception Score in case of GDA is relatively larger.

	Iteration			
Method	5000	10000	25000	50000
Ours	2.71 (0.28)	3.57 (0.26)	4.10 (0.35)	**4.68** (0.39)
GDA	2.80 (0.52)	3.56 (0.64)	4.28 (0.77)	4.51 (0.86)
OMD	1.60 (0.18)	1.80 (0.37)	1.73 (0.25)	1.96 (0.26)

Figure 16: GAN trained using GDA (with $k = 1$ discriminator steps). We repeated this simulation multiple times; here we display images generated from some of the resulting generators for GDA.

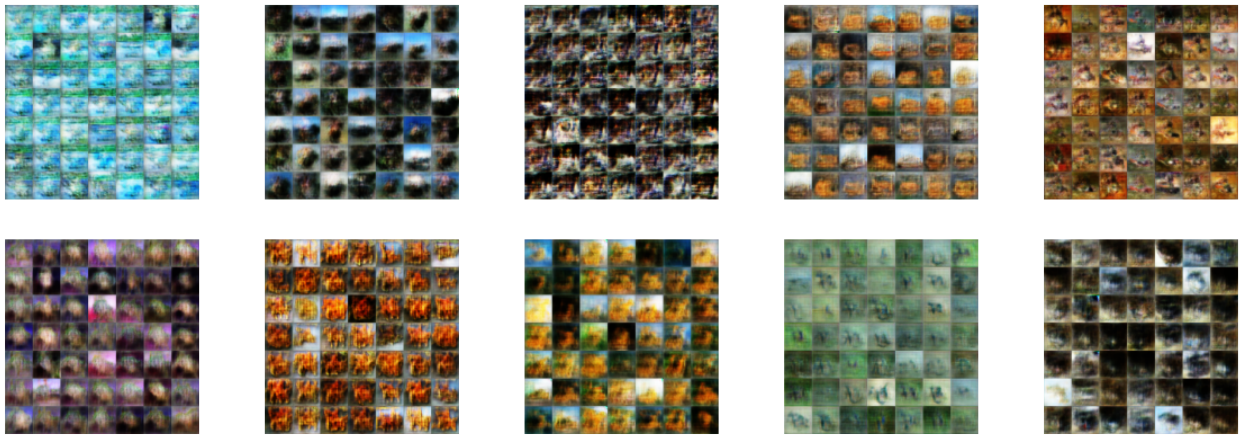

Figure 17: GAN trained using OMD. We repeated this simulation multiple times; here we display images generated from some of the resulting generators for OMD.

E Empirical Results for CIFAR-10 Dataset

This real-world dataset contains 60K color images from 10 classes. Previous works [5, 40, 53] have noted that it is challenging to detect mode collapse on CIFAR-10, visually or using standard metrics such as Inception Scores, because the modes are not well-separated. We use this dataset primarily to compare the scalability, quality, and stability of GANs in our framework obtained using our training algorithm.

For CIFAR-10, in addition to providing images generated by the GANs, we also report the Inception Scores [50] at different iterations. Inception Score is a standard heuristic measure for evaluating the quality of CIFAR-10 images and quantifies whether the generated images correspond to specific objects/classes, as well as, whether the GAN generates diverse images. A higher Inception Score is better, and the lowest possible Inception Score is 1.

Hyperparameters for CIFAR-10 simulations. For the CIFAR-10 simulations, we use a batch size of 128, with Adam learning rate of 0.0002 and hyperparameter $\beta_1 = 0.5$ for both the generator and discriminator gradients. Our code for the CIFAR-10 simulations is based on the code of Jason Brownlee [6], which originally used gradient descent ascent and ADAM gradients for training.

For the generator we use a neural network with input of size 100 and 4 hidden layers. The first hidden layer consists of a dense layer with $4,096$ parameters, followed by a leaky RELU layer, whose activations are reshaped into 246 4×4 feature maps. The feature maps are then upscaled to an output shape of 32 x 32 via three hidden layers of size 128 each consisting of a convolutional *Conv2DTranspose* layer followed by a leaky RELU layer, until the output layer where three filter maps (channels) are created. Each leaky RELU layer has "alpha" parameter 0.2.

For the discriminator, we use a neural network with input of size $32 \times 32 \times 3$ followed by 5 hidden layers. The first four hidden layers each consist of a convolutional *Conv2DTranspose* layer followed by a leaky RELU layer with "alpha" parameter 0.2. The first layer has size 64, the next two layers each have size 128, and the fourth layer has size 256. The output layer consists of a projection to 1 dimension with dropout regularization of 0.4 and sigmoid activation function.

Hardware. Our simulations on the CIFAR-10 dataset were performed on the above, and using one GPU with High frequency Intel Xeon E5-2686 v4 (Broadwell) processors, provided by AWS.

Results for CIFAR-10. We ran our algorithm (with $k = 1$ discriminator steps and acceptance rate $e^{-\frac{1}{\tau}} = \frac{1}{2}$) on CIFAR-10 for 20 repetitions and 50,000 iterations per repetition. We compare with GDA with $k = 1$ discriminator steps and OMD. For all algorithms, we compute the Inception Score every 500 iterations; Table 3 reports the Inception Scores at iteration 5000, 10000, 25000, and 50000, while Figure 14 provides the complete plot for Inception Score vs. training iterations. Sample images from all three algorithms are also provided in Figures 15, 16, 17.

The average Inception Score of GANs from both GDA and our algorithm are fairly close to each other, with the final mean Inception Score of 4.68 for our algorithm being somewhat higher than the final mean of 4.51 for GDA. However, the standard error of Inception Scores of GDA is much larger than of our algorithm. The relatively larger standard deviation of GDA is because

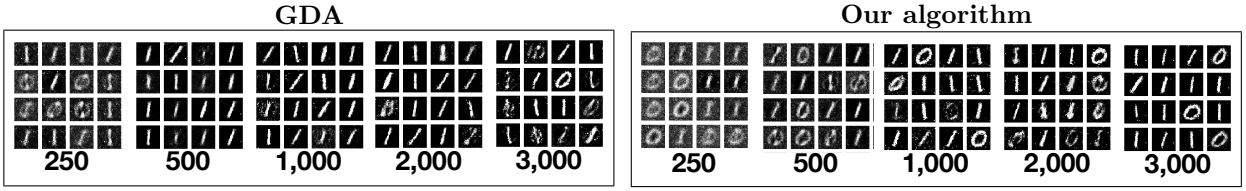

Figure 18: We trained a GAN using our algorithm on 0-1 MNIST for 30,000 iterations (with $k = 1$ discriminator steps and acceptance rate $e^{-\frac{1}{\tau}} = \frac{1}{5}$). We repeated this experiment 22 times for our algorithm and 13 times for GDA. Shown here are the images generated from one of these runs at various iterations for our algorithm (right) and GDA (left).

GDA, in certain runs, does not learn an appropriate distribution at all (Inception Score is close to 1 throughout training in this case), leading to a larger value of standard deviation. Visually, in these GDA runs, the GANs from GDA do not generate recognizable images (Figure 16, top-right image). For all other trials, the images generated by GDA have similar Inception Score (and similar quality) as the images generated by our algorithm. In other words, our algorithm seems to be more stable than GDA and returns GANs that generate high quality images in every repetition.

GANs trained using OMD attain much lower Inception Scores than our algorithm.[4] Moreover, the images generated by GANs trained using OMD have visually much lower quality than the images generated by GANs trained using our algorithm (Figure 17).

Evaluation on CIFAR-10 dataset shows that the GANs from our training algorithm can always generate good quality images; in comparison to OMD, the GANs trained using our algorithm generate higher quality images, while in comparison to GDA, it is relatively more stable.

Clock time per iteration. When training on CIFAR-10, our algorithm and GDA both took the same amount of time per iteration, 0.08 seconds, on the AWS GPU server.

We evaluate our algorithm on MNIST dataset as well, where it also learns to generate from multiple modes; the results are presented in Appendix F.

F Empirical Results for MNIST Dataset

This dataset consists of 60k images of hand-written digits [29]. We use two versions of this dataset: the full dataset and the dataset restricted to 0-1 digits.

Hyperparameters for MNIST simulations. For the MNIST simulations, we use a batch size of 128, with Adam learning rate of 0.0002 and hyperparameter $\beta_1 = 0.5$ for both the generator and discriminator gradients. Our code for the MNIST simulations is based on the code of Renu Khandelwal [25] and Rowel Atienza [3], which originally used gradient descent ascent and ADAM gradients for training.

For the generator we use a neural network with input of size 256 and 3 hidden layers, with leaky RELUS each with "alpha" parameter 0.2 and dropout regularization of 0.2 at each layer. The first

[4]We could not replicate the performance of OMD reported in [13], even with the implementation provided here - `https://github.com/vsyrgkanis/optimistic_GAN_training`.

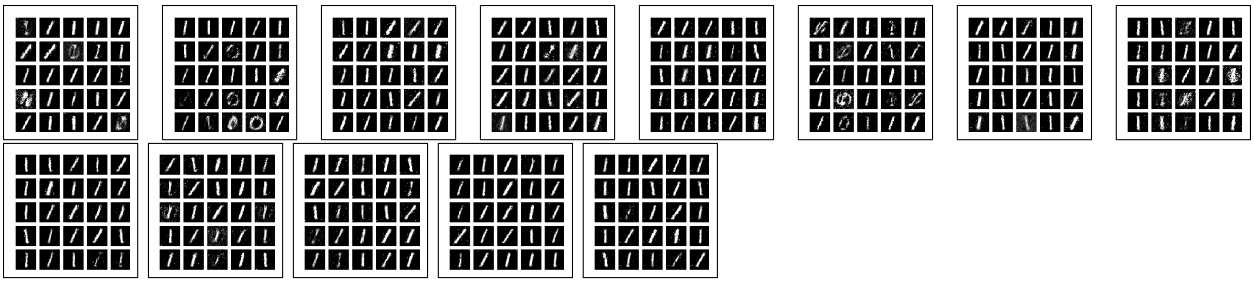

Figure 19: Images generated at the 1000'th iteration of the 13 runs of the GDA simulation mentioned in Figure 18. In 77% of the runs the generator seems to be generating only 1's at the 1000'th iteration.

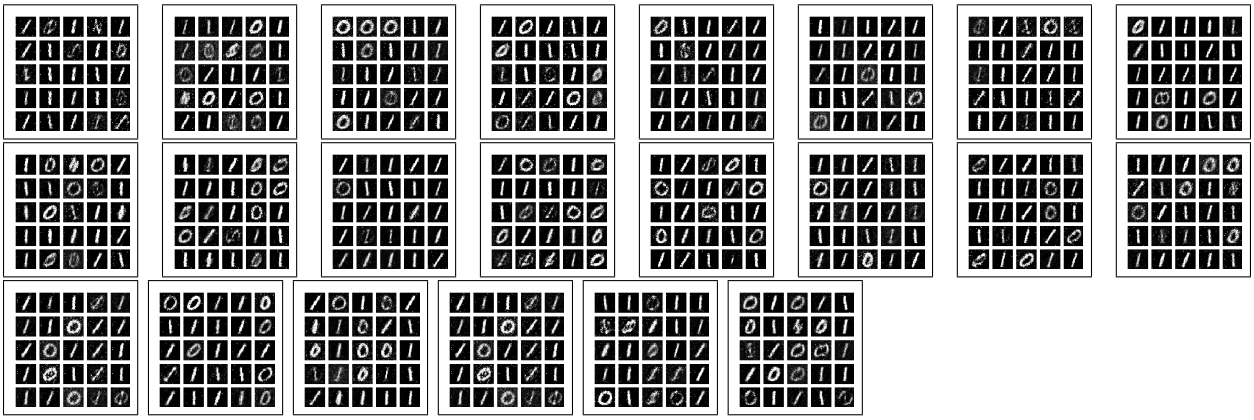

Figure 20: Images generated at the 1000'th iteration of each of the 22 runs of our algorithm for the simulation mentioned in Figure 18.

layer has size 256, the second layer has size 512, and the third layer has size 1024, followed by an output layer with hyperbolic tangent ("tanh") acvtivation.

For the discriminator we use a neural network with 3 hidden layers, and leaky RELUS each with "alpha" parameter 0.2, and dropout regularization of 0.3 (for the first two layers) and 0.2 (for the last layer). The first layer has size 1024, the second layer has size 512, the third layer has size 256, and the hidden layers are followed by a projection to 1 dimension with sigmoid activation (which is fed as input to the cross entropy loss function).

Results for 0-1 MNIST. We trained GANs using both GDA and our algorithm on the 0-1 MNIST dataset, and ran each algorithm for 3000 iterations (Figure 18). GDA seems to briefly generate shapes that look like a combination of 0's and 1's, then switches to generating only 1's, and then re-learns how to generate 0's. In contrast, our algorithm seems to learn how to generate both 0's and 1's early on and does not mode collapse to either digit. (See Figure 19 for images generated by all the runs of GDA, and Figure 20 for images generated by the GAN for all the runs of our algorithm.)

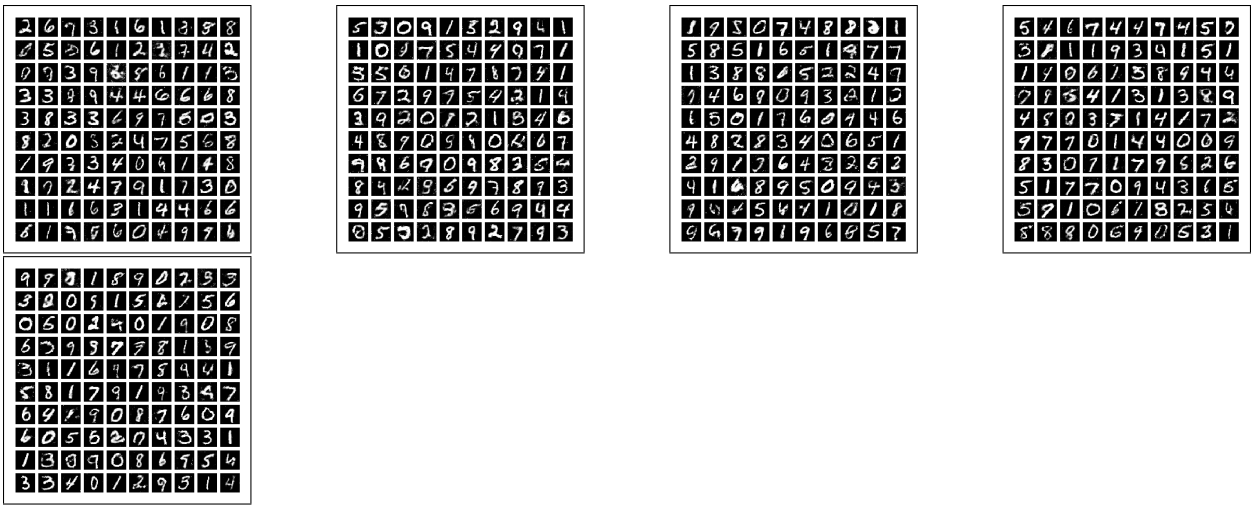

Figure 21: We ran our algorithm (with $k = 1$ discriminator steps and acceptance rate $e^{-\frac{1}{\tau}} = \frac{1}{5}$) on the full MNIST dataset for 39,000 iterations, and then plotted images generated from the resulting generator. We repeated this simulation five times; the generated images from each of the five runs are shown here.

Full MNIST. Next we evaluate the utility of our algorithm on the full MNIST dataset. We trained a GAN on the full MNIST dataset using our algorithm for 39,000 iterations (with $k = 1$ discriminator steps and acceptance rate $e^{-\frac{1}{\tau}} = \frac{1}{5}$). We ran this simulation five times; each time the GAN learned to generate all ten digits (see Fig. 21 for generated images).

G Randomized Acceptance Rule with Decreasing Temperature

In this section we give the simulations mentioned in the paragraph towards the beginning of Section 5, which discusses simplifications to our algorithm. We included these simulations to verify that our algorithm also works well when it is implemented using a randomized acceptance rule with a decreasing temperature schedule (Figure 22).

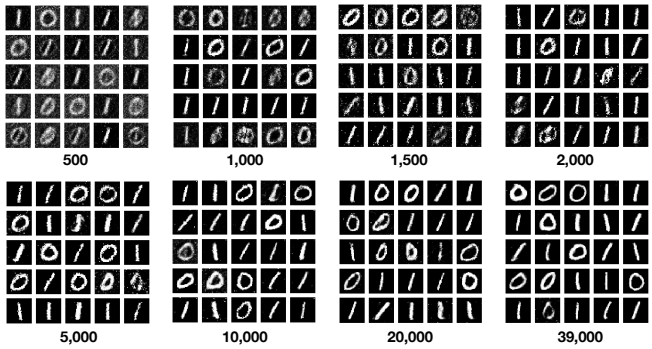

Figure 22: In this simulation we used a randomized accept/reject rule, with a decreasing temperature schedule. The algorithm was run for 39,000 iterations, with a temperature schedule of $e^{-\frac{1}{\tau_i}} = \frac{1}{4+e^{(i/20000)^2}}$. Proposed steps which decreased the computed value of the loss function were accepted with probability 1, and proposed steps which increased the computed value of the loss function were rejected with probability $\max(0, 1 - e^{-\frac{i}{\tau_1}})$ at each iteration i. We ran the simulation 5 times, and obtained similar results each time, with the generator learning both modes. In this figure, we plotted the generated images from one of the runs at various iterations, with the iteration number specified at the bottom of each figure (see also Figure 23 for results from the other four runs)

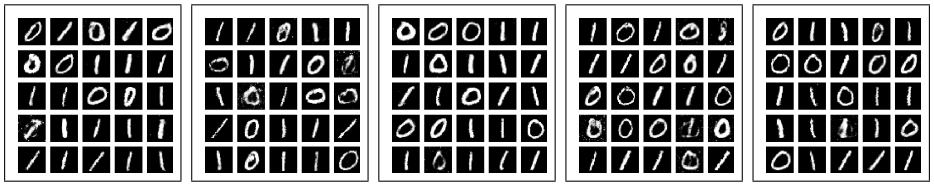

Figure 23: Images generated at the 39,000'th iteration of each of the 5 runs of our algorithm for the simulation mentioned in Figure 22 with a randomized acceptance rule with a temperature schedule of $e^{-\frac{1}{\tau_i}} = \frac{1}{4+e^{(i/20000)^2}}$.