Université de Montréal

Adversarial Games in Machine Learning: Challenges and Applications

par

Hugo Berard

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

August, 2022

Université de Montréal
Faculté des arts et des sciences

Cette thèse intitulée:

Adversarial Games in Machine Learning: Challenges and Applications

présentée par:

Hugo Berard

a été évaluée par un jury composé des personnes suivantes:

Ioannis Mitliagkas,	président-rapporteur
Pascal Vincent,	directeur de recherche
Simon Lacoste-Julien,	codirecteur
Margarida Carvalho,	membre du jury
Lillian J. Ratliff,	examinateur externe

Thèse acceptée le: ....

Résumé

L'apprentissage automatique repose pour un bon nombre de problèmes sur la minimisation d'une fonction de coût, pour ce faire il tire parti de la vaste littérature sur l'optimisation qui fournit des algorithmes et des garanties de convergences pour ce type de problèmes. Cependant récemment plusieurs modèles d'apprentissage automatique qui ne peuvent pas être formulé comme la minimisation d'un coût unique ont été propose, à la place ils nécessitent de définir un jeu entre plusieurs joueurs qui ont chaque leur propre objectif. Un de ces modèles sont les réseaux antagonistes génératifs (GANs). Ce modèle génératif formule un jeu entre deux réseaux de neurones, un générateur et un discriminateur, en essayant de tromper le discriminateur qui essaye de distinguer les vraies images des fausses, le générateur et le discriminateur s'améliore résultant en un équilibre de Nash, ou les images produites par le générateur sont indistinguable des vraies images. Malgré leur succès les GANs restent difficiles à entrainer à cause de la nature antagoniste du jeu, nécessitant de choisir les bons hyperparamètres et résultant souvent en une dynamique d'entrainement instable. Plusieurs techniques de régularisations ont été propose afin de stabiliser l'entrainement, dans cette thèse nous abordons ces instabilités sous l'angle d'un problème d'optimisation. Nous commençons par combler le fossé entre la littérature d'optimisation et les GANs, pour ce faire nous formulons GANs comme un problème d'inéquation variationnelle, et proposons de la littérature sur le sujet pour proposer des algorithmes qui convergent plus rapidement. Afin de mieux comprendre quels sont les défis de l'optimisation des jeux, nous proposons plusieurs outils afin d'analyser le paysage d'optimisation des GANs. En utilisant ces outils, nous montrons que des composantes rotationnelles sont présentes dans le voisinage des équilibres, nous observons également que les GANs convergent rarement vers un équilibre de Nash mais converge plutôt vers des équilibres stables locaux (LSSP). Inspirer par le succès des GANs nous proposons pour finir, une nouvelle famille de jeux que nous appelons *adversarial example games* qui consiste à entrainer simultanément un générateur et un critique, le générateur cherchant à perturber les exemples afin d'induire en erreur le critique, le critique cherchant à être robuste aux perturbations. Nous montrons qu'à l'équilibre de ce jeu, le générateur est capable de générer des perturbations qui transfèrent à toute une famille de modèles.

Abstract

Many machine learning (ML) problems can be formulated as minimization problems, with a large optimization literature that provides algorithms and guarantees to solve this type of problems. However, recently some ML problems have been proposed that cannot be formulated as minimization problems but instead require to define a game between several players where each player has a different objective. A successful application of such games in ML are generative adversarial networks (GANs), where generative modeling is formulated as a game between a generator and a discriminator, where the goal of the generator is to fool the discriminator, while the discriminator tries to distinguish between fake and real samples. However due to the adversarial nature of the game, GANs are notoriously hard to train, requiring careful fine-tuning of the hyper-parameters and leading to unstable training. While regularization techniques have been proposed to stabilize training, we propose in this thesis to look at these instabilities from an optimization perspective. We start by bridging the gap between the machine learning and optimization literature by casting GANs as an instance of the Variational Inequality Problem (VIP), and leverage the large literature on VIP to derive more efficient and stable algorithms to train GANs. To better understand what are the challenges of training GANs, we then propose tools to study the optimization landscape of GANs. Using these tools we show that GANs do suffer from rotation around their equilibrium, and that they do not converge to Nash-Equilibria. Finally inspired by the success of GANs to generate images, we propose a new type of games called Adversarial Example Games that are able to generate adversarial examples that transfer across different models and architectures.

Keywords - Mots-clés

Keywords. machine learning, neural networks, generative modeling, generative adversarial networks (GAN), adversarial attacks, adversarial training, game theory, optimization, variational inequality, Nash-equilibrium, extragradient, visualization

Mots-clés. apprentissage automatique, réseaux de neurones, modèles génératifs, réseaux antagonistes génératifs, exemples antagonistes, théorie des jeux, optimisation, inéquation variationelle, équilibre de Nash, extragradient, visualisation

Contents

List of Tables

List of Figures

List of acronyms and abbreviations

e.g.	*exempli gratia* [for instance]
i.e.	*ide est* [that is]
iff	if and only if
resp.	respectively
ML	Machine Learning
DL	Deep Learning
DNN	Deep Neural Networks
KL	Kullback-Leibler
MMD	Maximum Mean Discrepancy
RKHS	Reproducing Kernel Hilbert Space
ERM	Empirical Risk Minimization
MLE	Maximum Likelihood Estimation
CNN	Convolutional Neural Network
VAE	Variational Auto-Encoder
NLL	Negative Log-Likelihood
GANs	Generative Adversarial Networks
PGD	Projected Gradient Descent
LSSP	Locally Stable Stationary Point
NE	Nash Equilibrium
SGD	Stochastic Gradient Descent
SGDA	Stochastic Gradient Descent-Ascent
VIP	Variational Inequality Problem
AEG	Adversarial Example Game
HGD	Hamiltonian Gradient Descent

Notation

The set of real numbers	$\mathbb{R}$
The set of complex numbers	$\mathbb{C}$
The real and imaginary part of $z \in \mathbb{C}$	$\Re(z)$ and $\Im(z)$
Scalars are lower-case letters	λ
Vectors are lower-case bold letters	$\boldsymbol{\theta}$
Matrices are upper-case bold letters.......	$\boldsymbol{A}$
Operators are upper-case letters	F
The spectrum of a squared matrix $\boldsymbol{A}$	$\mathrm{Sp}(\boldsymbol{A})$
The spectral radius of a squared matrix $\boldsymbol{A}$ ...	$\rho(A)$
The largest and the smallest singular values of $\boldsymbol{A}$	$\sigma_{\min}(\boldsymbol{A})$ and $\sigma_{\max}(\boldsymbol{A})$
The identity matrix of $\mathbb{R}^{d\times d}$	$\boldsymbol{I}_d$
Standard asymptotic notations.........	$\mathcal{O}$, Ω and Θ

1 Introduction

The first transistor was invented in 1947 by Bardeen and Brattain [1948] paving the way for modern electronics and computers. Since then progress in the semiconductor industry has lead to significant improvement in transistor size and density, enabling to drastically increase the number of transistor in a chip. The most recent chips to date are manufactured using extreme ultraviolet lithography (ULV) [Wu and Kumar, 2007] allowing for a density of over 100 millions transistors per mm^2. Those major innovations have enabled two things: 1) this significantly decreased the cost and the size of electronic circuits and sensors making them widely available to collect and store data. 2) This greatly increased the computational power available to us, enabling us to run much more complex and expensive algorithms. This combination of large amounts of readily available data with large computational power has propelled the field of Machine Learning (ML) forward.

Machine Learning is the science of using data and algorithms to learn models and make predictions, by using data we are able to infer a model that can then be used to make predictions. One of the first descriptions of using a technique to infer a mathematical model from observations was proposed by Legendre [1806] to predict the orbits of comets by fitting a linear model with least squares regression. Since then ML techniques have greatly improved enabling a wide range of applications that continues to grow. Among the many successful applications of ML, we can mention spam filtering [Guzella and Caminhas, 2009], image classification [Krizhevsky et al., 2012], image generation [Goodfellow et al., 2014], language modeling [Brown et al., 2020], and achieving superhuman performance at Go [Silver et al., 2016]. Furthermore, recent applications to other scientific domains have also shown great promise with the potential to accelerate scientific discoveries in several fields. For example Jumper et al. [2021] was recently able to predict the structure of proteins with high accuracy enabling potential important discovery in biology or accelerating drug research. This potential for accelerating scientific discoveries by using ML has also been recently used for material science [Schmidt et al., 2019]. A recent example is the Open Catalyst challenge [Zitnick et al., 2020] where researchers are trying to find better catalysts to produce hydrogen more efficiently and help countries move away from fossil fuels by providing clean alternatives that do not emit green house gases (GHG) in the atmosphere, in order to respect the Paris Agreement and limit global warming to 2°C above pre-industrial levels.

A lot of the progress made relies on deep neural networks (DNN) [LeCun et al., 2015a], a class of non-convex models with a very large number of parameters which can approximate any functions [Hornik et al., 1989]. Furthermore the gradient of such models can be efficiently computed using the back-propagation algorithms [Rumelhart et al., 1986], and can thus be optimized by using gradient methods [Ruder, 2016]. Such models are usually trained by minimizing a single objective function by using gradient methods.

However some problems cannot be formulated as minimization problems. For example Goodfellow et al. [2014] proposed generative adversarial network (GAN), a generative model that combines two neural networks: a generator that tries to generate data points, and a discriminator that tries to classify if a data point comes from the generator or the dataset. This problem is formulated as a game between the generator and the discriminator, where the generator is trying to fool the discriminator, where the generator and discriminator are trying to optimize different objectives that are in competition with one another. Thus this problem cannot be formulated as a single objective minimization problem. GANs have been very successfully used to generate high resolution images [Brock et al., 2019] showing that using game formulations can outperform standard minimization approaches. Despite their success, GANs are challenging to train due the min-max nature of the game they are trying to solve. Successful training of GANs thus requires a careful choice of hyperparameters and the use of regularization techniques [Gulrajani et al., 2017, Miyato et al., 2018]. In this thesis we look at those instabilities during training by adopting an optimization perspective, we analyze where the instabilities come from and how to reduce those instabilities by using algorithms that are more suited to game optimization.

Other type of ML problems can also benefit from adopting a game formulation. Szegedy et al. [2014] showed that adding a small amount of noise to the input of a DNN can fool the classifier into predicting the wrong label for the input, such perturbations are called adversarial attacks. To fix this problem, Madry et al. [2018] proposed to train the models using a robust optimization formulation which involves adding a maximization over all perturbations inside the objective function. We propose to formulate this problem as a game between a generator and a critic. We show that using this approach the adversarial attacks generated can successfully transfer to different models.

Overall this thesis looks at how to leverage game formulations in machine learning applications, and tries to address some of the challenges of game optimization.

1 Overview of Thesis

Beyond this introduction and the conclusion, this thesis includes a background section that presents the key notions necessary to understand the rest of this thesis, as well as three contributions that each correspond to a research paper which was published in a conference. The contributions are organized as follows. The first contribution presents a connection between variational inequalities and GANs, showing some limitations of current algorithms and proposing new algorithms to train GANs. The second contribution looks at the optimization landscape of GANs through empirical observations. Finally the third contribution, presents a new application of games to the generation of adversarial attacks.

1.1 Optimization of GANs

GANs are notoriously hard to train, being very unstable and requiring a lot of fine tuning of the hyper-parameters. Several works looked at those instabilities from a divergence perspective [Arjovsky et al., 2017, Nowozin et al., 2016]. A few works also looked at such instabilities from an optimization perspective [Mescheder et al., 2017, Nagarajan and Kolter, 2017]. However no formal connection was made between GANs and the relevant optimization literature. In our first contribution we framed GANs as an instance of the variational inequality problem. By making this connection we were able to provide insight in the instability of GANs, we showed that in bilinear games Stochastic Gradient Descent-Ascent (SGDA) does not converge. Making this connection also enabled us to bring techniques from the variational inequality literature to improve the training in GANs. In particular by combining Adam with averaging and an extrapolation step, we proposed a new algorithm called ExtraAdam. We showed that this new algorithm consistently outperformed existing algorithms in several experiments. Averaging was also shown to be beneficial when used with other algorithms such as Adam.

The optimization literature usually assumes games are monotone to be able to derive convergence guarantees. However GANs are highly non-monotone, thus it is not clear whether analysis in the monotone case can provide insight into the behavior of GANs. In particular, while the presence of rotation is highly problematic from a theoretical point of view as it leads to convergence problem for SGDA for example, it is not clear if GANs exhibit such properties in practice. The second contribution tries to answer this question through empirical observations. We first provide a formal definition of the notion of the rotation in games by relating it to the imaginary part of the eigenvalues of the Jacobian of the game. We then propose two different visualization techniques to try to quantify the amount of rotation and other properties around a stationary point. Using those visualization techniques we studied the optimization landscape of GANs which enabled us to make several observations. We first noticed that contrary to popular beliefs GANs rarely converge to Nash-Equilibrium in practice but instead often converge to locally stable stationary points (LSSP). We further show that GANs while not being fully rotational, exhibit some amount of rotations that could explain why they are unstable.

1.2 Adversarial Attacks

In the third and last contribution we look at the problem of adversarial attacks. We propose a new framework to formulate adversarial attacks as a game, where the attacker is also a deep neural networks that crafts the attack. By formulating this problem as a game, the attacker is able to attack a class of functions instead of a specific function. We showed experimentally that this is indeed the case and that using this framework the attacks we generate are able to transfer to a wide range of neural networks.

2 Excluded Research

During my PhD I also contributed to the following works which will be excluded from this thesis to keep the manuscript consistent and succinct:

- Parametric Adversarial Divergences are Good Losses for Generative Modeling [Huang et al., 2017a]
- Stochastic hamiltonian gradient methods for smooth games [Loizou et al., 2020]
- Online Adversarial Attacks [Mladenovic et al., 2022]
- Stochastic gradient descent-ascent and consensus optimization for smooth games: Convergence analysis under expected co-coercivity [Loizou et al., 2021]
- Stochastic gradient descent-ascent: Unified theory and new efficient methods [Beznosikov et al., 2022]
- Stochastic extragradient: General analysis and improved rates [Gorbunov et al., 2022]

2 Background

1 Statistical Decision Theory

Let us consider the universe to be non-deterministic. We can represent it as a distribution $P_{\mathcal{D}}$. This distribution represents the fundamental truth that governs the universe which is not known to us. While this distribution is unknown to us, we can formulate different hypothesis $h \in \mathcal{H}$, where $\mathcal{H}$ is the set of hypothesis we consider. Our goal is to find the hypothesis $h \in \mathcal{H}$ that is as close as possible to the ground-truth distribution of the universe $P_{\mathcal{D}}$. To find such an hypothesis, we can use the observations we make about the universe by sampling from the distribution of the universe $P_{\mathcal{D}}$. We will denote those observations $\mathcal{D} \sim P_{\mathcal{D}}$. We also define an estimator $\delta \in \mathcal{A}$, a function that given some observations selects an hypothesis, we denote this estimator by $\delta : \Omega_{\mathcal{D}} \rightarrow \mathcal{H}$. To measure the quality of an hypothesis we define a loss function $\mathcal{L}(h, P_{\mathcal{D}}) \in \mathbb{R}^+$, it represents the error of choosing the hypothesis h instead of the ground-truth $P_{\mathcal{D}}$. To compare two different estimators δ_1 and δ_2, and decide if one is better than the other at selecting the best hypothesis, we can use the frequentist risk:

$$R(\delta, P_{\mathcal{D}}) = \mathbb{E}_{D \sim P_{\mathcal{D}}}[\mathcal{L}(\delta(D), P_{\mathcal{D}})] \tag{1.1}$$

Note that in practice this quantity cannot be evaluated since we do not know $P_{\mathcal{D}}$, this is thus a purely theoretical quantity. Also note that this quantity depends on $P_{\mathcal{D}}$, thus an estimator δ_1 can outperform another estimator δ_2 on $P_{\mathcal{D}}$ but might perform worse on a different universe with distribution $P'_{\mathcal{D}}$.

1.1 Statistical Divergence

As mentioned in the previous section to be able to compare hypothesis and measure how close an hypothesis is to the ground-truth we need to define a loss function $\mathcal{L}$.

In this section we will look at loss functions defined as distance between probability distribution. There is a large literature about statistical distances and their properties, here we will present a few statistical distances that are often used in machine learning.

Let us first introduce some notations. Let $\mathcal{X}$ be a compact metric space, and let Σ be the set of all Borel subsets of $\mathcal{X}$. Let $\text{Prob}(\mathcal{X})$ be the space of probability measures defined on $\mathcal{X}$, we can now define distances between probability distributions $\mathbb{P}, \mathbb{Q} \in \text{Prob}(\mathcal{X})$.

KL-divergence [**Kullback and Leibler, 1951**]. One of the most used and popular statistical distance is the Kullback–Leibler divergence:

$$\text{KL}(\mathbb{P}, \mathbb{Q}) = \int_{\mathcal{X}} \log\left(\frac{p(x)}{q(x)}\right) p(x)dx \tag{1.2}$$

The KL is not symmetric and can be infinite if the supports of $\mathbb{P}$ and $\mathbb{Q}$ do not fully overlap, i.e. $\exists x \in \mathcal{X}$ such that $q(x) = 0$ and $p(x) > 0$.

***f*-divergence family** [**Rényi et al., 1961**]. The f-divergence family is a generalization of Kullback-Leibler divergence:

$$D_f(\mathbb{P}, \mathbb{Q}) = \int_{\mathcal{X}} f\left(\frac{p(x)}{q(x)}\right) p(x)dx \tag{1.3}$$

where the function $f : \mathbb{R}^+ \to \mathbb{R}$ is convex, lower-semicontinuous and satisfies $f(1) = 0$.

Wasserstein-*p* distance [**Kantorovich, 1960**]. Another type of divergence is the Wasserstein-p distance:

$$W_p(\mathbb{P}, \mathbb{Q}) = \left(\inf_{\gamma \in \Gamma(\mathbb{P}, \mathbb{Q})} \int d(x, y)^p d\gamma(x, y)\right)^{\frac{1}{p}} \tag{1.4}$$

where $\Gamma(\mathbb{P}, \mathbb{Q})$ is the set of all joint distributions such that with marginals $\mathbb{P}$ and $\mathbb{Q}$, and where d is a metric on $\mathcal{X}$.

The Wasserstein distance was first proposed to solve optimal transport problems, see Villani [2009] for a comprehensive introduction to optimal transport and the Wasserstein distance.

Maximum Mean Discrepancy (MMD) [**Gretton et al., 2012**]. This divergence is also known as an integral probability metric [Müller, 1997], and is a generalization of the Wasserstein metric:

$$\mathrm{MMD}(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathcal{F}} \Big(\int f(x) dp(x) - \int f(y) dq(y) \Big) \tag{1.5}$$

where $\mathcal{F}$ is a class of functions $f : \mathcal{X} \to \mathbb{R}$. Note that MMD usually refers to the case where $\mathcal{F}$ is chosen to be the unit ball in a reproducing kernel Hilbert space (RKHS).

Those divergences have different properties. For a comparison refer to Huang et al. [2017a] which provides a in-depth analysis of different choice of divergence.

1.2 Empirical Risk Minimization

Now that we introduced several notions of distance, we can look at different estimators.

The first estimator we will present is called the empirical risk minimizer (ERM), and is one of the core principles of machine learning.

One of the most common use case for machine learning, is when we want to understand the relationship between two variables. Let us assume we collect some pair of observations $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$. We want to understand the relation between x and y, in particular we want to be able to predict y given x. We thus want to find the best function $f : \mathcal{X} \to \mathcal{Y}$, where the set of functions $f \in \mathcal{F}$ is our hypothesis class. Our goal is to find the best possible hypothesis f^*:

$$f^* = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{D} \sim P_{\mathcal{D}}} [\ell(f(x), y)] \tag{1.6}$$

where here $\ell(\hat{y}, y)$ is a function that represents the error of making prediction $\hat{y}$ instead of y.

However we don not know $P_{\mathcal{D}}$ and usually only have access to a finite subset. An alternative is thus to solve the following problem:

$$\hat{f}_{\mathrm{ERM}}(\mathcal{D}) = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \tag{1.7}$$

This problem is called empirical risk minimization, and the estimator we derive from solving this problem is called the empirical risk minimizer. In practice we often only have access to a finite-sample dataset, and we will thus often rely on empirical risk minimization to get an estimator of f^*.

1.3 Maximum Likelihood Estimation

Sometimes instead of learning a function we are interested in approximating some distribution $\mathbb{P}$ with some distribution $\mathbb{Q}$, we can formulate this problem like this:

$$\min_{\mathbb{Q}\in\mathcal{P}} \mathrm{D}(\mathbb{P},\mathbb{Q}) \tag{1.8}$$

Where $\mathrm{D} : \mathcal{P} \times \mathcal{P} \to \mathbb{R}^+$ is a function that measures the distance between two distributions. A divergence that is commonly used is the KL divergence (1.2).

Let us now go back to the case where we want to find a distribution $\mathbb{Q}$ that approximates a distribution $\mathbb{P}$. From far the KL divergence seems to be intractable since it involves the knowledge of $p(\mathrm{x})$. However by using a clever decomposition we can show that we can minimize the KL divergence and that minimizing the KL divergence only requires to be able to sample from $\mathbb{P}$:

$$\begin{aligned}\min_{\mathbb{Q}\in P} D_{\mathrm{KL}}(\mathbb{P},\mathbb{Q}) &= \min_{\mathbb{Q}\in\mathcal{P}} \mathbb{E}_{\mathrm{x}\sim\mathbb{P}}\left[\log\left(\frac{p(\mathrm{x})}{q(\mathrm{x})}\right)\right] &(1.9)\\ &= \min_{\mathbb{Q}\in\mathcal{P}} \mathbb{E}_{\mathrm{x}\sim\mathbb{P}}[\log p(\mathrm{x})] - \mathbb{E}_{\mathrm{x}\sim P}[\log q(\mathrm{x})] &(1.10)\end{aligned}$$

Since the first term is a constant, we can ignore it and it doesn't change the solution of the minimization problem:

$$\min_{\mathbb{Q}\in\mathcal{P}} D_{\mathrm{KL}}(\mathbb{P},\mathbb{Q}) = \max_{\mathbb{Q}\in\mathcal{P}} \mathbb{E}_{\mathrm{x}\sim\mathbb{P}}[\log q(\mathrm{x})] + c \tag{1.11}$$

We usually only have access to a finite sample dataset of sP, making this problem impossible to solve. However, we can apply the ERM principle from the previous section and replace the expectation by a finite sum, the problem thus becomes:

$$\max_{\mathbb{Q}\in\mathcal{P}} \frac{1}{n}\sum_{i=1}^{n} \log q(\mathrm{x}_i) = \max_{\mathbb{Q}\in\mathcal{P}} \log \prod_{i=1}^{n} q(\mathrm{x}_i) = \max_{\mathbb{Q}\in\mathcal{P}} \log q(\mathrm{x}) \tag{1.12}$$

where $\log q(\mathrm{x})$ is called the log likelihood and $q(\mathrm{x}) = q(\mathrm{x}_1, \cdots, \mathrm{x}_n) = \prod_{i=1}^n q(\mathrm{x}_i)$.

We call this estimator, the Maximum Likelihood Estimator (MLE):

$$\hat{\mathbb{Q}}_{\mathrm{MLE}} = \arg\max_{\mathbb{Q}\in\mathcal{P}} \log q(\mathrm{x}) \tag{1.13}$$

Maximum likelihood estimation and ERM are at the core of many machine learning algorithms. However these relatively simple principles raise many questions: how do we solve the underlying minimization or maximization problem? What hypothesis class should we consider for f and Q? And, what loss should we use for ERM? We will introduce some popular choice in the next sections.

2 Linear models

Let us consider the ERM problem from the previous section, one of the simplest hypothesis class we can consider is the class of linear models $\mathrm{LM} = \{f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T\boldsymbol{x}, \boldsymbol{\theta} \in \mathbb{R}^d\}$. Such models are extremely popular due to their simplicity. If we decide to use the square loss $\ell(\hat{y}, y) = (y - \hat{y})^2$, we get the linear least squares problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n (\boldsymbol{\theta}^T\boldsymbol{x}_i - y_i)^2 \tag{2.1}$$

By using matrix notation, we can define, $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ with $\boldsymbol{X}_i = \boldsymbol{x}_i^T$, and $\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$,

we can rewrite the previous problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \|\boldsymbol{X}\boldsymbol{\theta} - y\|^2 \tag{2.2}$$

To find $\hat{w}$ we can compute the gradient:

$$\nabla_\theta \|\boldsymbol{X}\boldsymbol{\theta} - y\|^2 = 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{X}^T y \tag{2.3}$$

Since we want to find the minimum, we want the gradient to be zero, we thus want to solve:

$$2\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\theta}} - 2\boldsymbol{X}^T y = 0 \tag{2.4}$$

If the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is invertible the solution is unique:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T y \tag{2.5}$$

While this model is very interesting due to simplicity, it suffers from several limitations. First while it is adapted to the case where y is a real value variable, we are often confronted to problems where y might be a categorical or binary variable. The class of linear functions is also very restricted and we might be interested more complex functions. We will see how to improve over linear models next.

2.1 Probabilistic Perspective

We have approached the linear least squares problem from the ERM perspective. Here we will show how we can adopt a probabilistic perspective and relate it to MLE.

Let us consider that y is a random variable that follows some unknown conditional distribution $p(\mathrm{y}|\mathbf{x})$ that we want to approximate. To approximate it we consider the following distribution:

$$q(\mathrm{y}|\mathbf{x};\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}^T\mathbf{x}, \sigma^2) \tag{2.6}$$

We thus model y as a gaussian random variable with a mean that linearly depends on $\mathbf{x}$. We can also write this in the following alternative fashion:

$$\mathrm{y} = \boldsymbol{\theta}^T\mathbf{x} + \epsilon \tag{2.7}$$

where $\epsilon = \mathcal{N}(0, \sigma^2)$ is a random variable independent of $\mathbf{x}$.

We want to find the maximum likelihood estimator for this problem:

$$\max_{\boldsymbol{\theta}\in\mathbb{R}^d} \log q(\mathrm{y}|\mathbf{x};\boldsymbol{\theta}) \tag{2.8}$$

where:

$$\begin{aligned}
\log q(\mathrm{y}|\mathbf{x};\boldsymbol{\theta}) &= \log q(\mathrm{y}_i, \cdots, \mathrm{y}_n|\mathbf{x}_1, \cdots, \mathbf{x}_n;\boldsymbol{\theta}) = \log \prod_{i=1}^n \log q(\mathrm{y}_i|\mathbf{x}_i;\boldsymbol{\theta}) && (2.9)\\
&= \sum_{i=1}^n q(\mathrm{y}_i|\mathbf{x}_i;\boldsymbol{\theta}) = \sum_{i=1}^n \left[-\frac{\left(y_i - \boldsymbol{\theta}^\top \boldsymbol{x}_i\right)^2}{2\sigma^2} - \frac{1}{2}\log\left(2\pi\sigma^2\right) \right] && (2.10)\\
&= -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2}\sum_{i=1}^n \frac{\left(y_i - \boldsymbol{\theta}^\top \boldsymbol{x}_i\right)^2}{\sigma^2} && (2.11)
\end{aligned}$$

Since the first term is a constant we can rewrite the MLE problem as:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{\theta}^T\boldsymbol{x}_i - y_i)^2 \tag{2.12}$$

which is the exact same problem as the linear least square problem. This shows that a lot of methods can both be derived from the ERM principal and from the MLE principle.

2.2 Logistic Regression

In the model we have presented above the variable y we want to predict takes real values. However in practice we are often interested in problems where $y \in \{0, 1\}$. Such problems are called binary classification problem. One simple but popular model for this setting is called logistic regression. In logistic regression we choose to model y has a Bernoulli random variable, such that:

$$\begin{aligned} q(\mathrm{y}=1|\mathbf{x};\boldsymbol{\theta}) &= \frac{1}{1+\exp(-f_{\boldsymbol{\theta}}(\boldsymbol{x}))} && (2.13)\\ &= 1 - q(\mathrm{y}=0|\mathbf{x}) && (2.14)\\ &= \sigma(f(\boldsymbol{x})) && (2.15)\end{aligned}$$

Where σ is often called the sigmoid function. The sigmoid function has two interesting properties:

$$\sigma(-x) = 1 - \sigma(x) \tag{2.16}$$

$$\sigma'(x) = \sigma(x)\sigma(-x) = \sigma(x)(1-\sigma(x)) \tag{2.17}$$

Similarly to the previous section we will consider linear functions f of the form $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T\boldsymbol{x}$. We can compute the log likelihood of the corresponding model:

$$\begin{aligned} \log q(\mathrm{y}|\mathbf{x};\boldsymbol{\theta}) &= \sum_{i=1}^{n}\log q(\mathrm{y}=y_i|\mathbf{x}_i;\boldsymbol{\theta}) && (2.18)\\ &= \sum_{i=1}^{n} y_i\log q(\mathrm{y}=1|\mathbf{x}_i;\boldsymbol{\theta}) + (1-y_i)\log q(\mathrm{y}=0|\mathbf{x}_i;\boldsymbol{\theta}) && (2.19)\\ &= \sum_{i=1}^{n} y_i\log\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + (1-y_i)\log(1-\sigma(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) && (2.20)\\ &= \sum_{i=1}^{n}\mathrm{BCE}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) && (2.21)\end{aligned}$$

The function BCE is often called Binary Cross Entropy and is often used in machine learning.

Contrary to the linear least square problem, there is no closed form solution for this problem. We will in future sections show how we can find approximate solutions efficiently, using iterative methods such as gradient descent.

3 Deep Neural Networks

The choice of function $f_{\boldsymbol{\theta}}$ is very important in ML. While computing solutions for problems involving linear models can often be done efficiently, such models are not very expressive and can only model simple relations between variables. The world is often very non-linear, we thus need a way to express more complex functions that can captures those non-linearities. A class of models which is very popular and is able to model complex relationship between variables is deep neural networks (DNN).

DNN are composed of several layers where the different layers are separated by non-linear activation functions. We can represent such models as a composition of functions:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = f_L \circ \cdots f_2 \circ f_1(\boldsymbol{x}) \tag{3.1}$$

Each layer will take as input the output of the previous layer and transform it. The output of each layer is called the activations and the final layer outputs the prediction.

The simplest type of DNN is the multi-layer perceptron (MLP), where each layer applies a linear transformation to the input followed by a non-linearity where $f_l(\boldsymbol{x}) = \sigma(\boldsymbol{\theta}_l \boldsymbol{x})$ is the activation and $\boldsymbol{\theta}_l \in \mathbb{R}^{d_l \times d_{l-1}}$ is the parameter of the l-th layer, σ is a non-linear function, a popular choice is the ReLU function $\sigma(x) = \max(0, x)$. This type of model is highly expressive and can approximate any functions if the model is sufficiently large, i.e. $L \geq 2$ and d_l is large enough [Hornik et al., 1989].

Another very popular type of DNN which are used to process images are Convolutional neural networks (CNN). They use convolutions operation to allow for parameter sharing and thus decrease the number of parameters while remaining highly expressive. To train very deep networks, He et al. [2016] proposed to use residual connections, allowing to train deeper networks with up to 152 layers. Recently, attention mechanisms have been shown to greatly improve the performance of DNN on a variety of tasks. Vaswani et al. [2017] first proposed the Transformer for language modeling. A variant of Transformer networks was proposed by Dosovitskiy et al. [2021] for image classification and other computer vision application.

One of the main advantages of DNN is that we can compute the gradient of such models efficiently using the back-propagation algorithm, which enable us to use gradient methods to solve the minimization problem and find a solution to (1.7). For further details about deep learning please refer [Goodfellow et al., 2016].

4 Generative models

Sometimes we want to be able to model distributions by approximating the distribution $P_{\mathcal{D}}$ of the data by a parametric distribution P_θ, where the distribution P_θ depends on some parameters θ that we want to learn. We can formulate this problem as minimizing the divergence between $P_{\mathcal{D}}$ and P_θ:

$$\min_\theta \mathrm{Div}(P_\theta, P_{\mathcal{D}}) \tag{4.1}$$

Here Div is a statistical divergence that measures a distance between two distributions as described in Section 1.1.

Several methods have been proposed to solve such problems and generate images. In this section, we present variational auto-encoders (VAE) [Kingma and Welling, 2014], and generative adversarial networks (GANs) [Goodfellow et al., 2015a]. Other models have also been proposed such as autoregressive models [Van Oord et al., 2016], and diffusion models [Ho et al., 2020]. Recently some models have achieved impressive resulst in image generation, in particular Ramesh et al. [2022] proposed DALL-E-2 a generative model that can generate realistic images based on text queries with impressive results.

4.1 Variatonal Auto-encoder (VAE)

Variational auto-encoder were first porposed by [Kingma and Welling, 2014]. VAE are a latent variable model that use an encoder-decoder architecture, and are optimizing a lower-bound on the negative log-likelihood (NLL).

We first introduce some random variables $z \sim P_z$, such that we can write the distribution p_θ:

$$p_\theta(x) = \mathbb{E}_{z \sim P_z}[p_\theta(x|z)] \tag{4.2}$$

Unfortunately, directly computing the likelihood of such models is intractable. However we can derive a lower bound on the log-likelihood, using Jensen-Inequality and by introducing a new distribution $q(z|x)$ that we call the approximate posterior.

We first introduce the approximate posterior by rewriting the likelihood as:

$$p_\theta(x) = \mathbb{E}_{p(z)}[p_\theta(x|z)] = \mathbb{E}_{q(z|x)}\left[\frac{p_\theta(x|z)p(z)}{q(z|x)}\right] \tag{4.3}$$

Then using the Jensen-Inequality, we can derive a lower-bound on the log-likelihood:

$$\log p_\theta(x) \geq \mathbb{E}_{q(z|x)}\left[\log \frac{p_\theta(x|z)p(z)}{q(z|x)}\right] = \mathbb{E}_{q(z|x)}[\log p_\theta(x|z)] - KL(q(z|x)|p(z)) \tag{4.4}$$

The first term is often referred to as the reconstruction error and the second term as the regularization term.

In practice both $p_\theta(x|z)$ and $q(z|x)$ are parameterized by deep neural networks, and we can thus use SGD to solve the following maximization problem:

$$\max_{\theta,\phi} \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x)|p(z)) \tag{4.5}$$

Several extensions of VAE have been proposed in the literature, such as VQ-VAE Van Den Oord et al. [2017] which uses discrete latent variables.

4.2 Generative Adversarial Networks

Generative adversarial Networks are a type of generative model proposed by Goodfellow et al. [2014] that defines a game between two neural networks, a generator and a discriminator, the generator tries to fool the discriminator while the discriminator tries to distinguish between fake and real images. They show that this game has a Nash-Equilibrium and that at this equilibrium the samples generated by the generator are indistinguishable from the real samples. Radford et al. [2016] were the first to successfully train a GAN to generate 64x64 images by using convolutional neural networks. However GANs remained challenging to train, Arjovsky and Bottou [2017] adopted a divergence perspective and showed that the game was not well defined leading to unstable training. In Arjovsky et al. [2017], they derive a new game from the dual formulation of the Wasserstein metric which requires the discriminator to be Lipschitz. Gulrajani et al. [2017], Miyato et al. [2018] propose different type of regularization to force the discriminator to be Lipschitz. To stabilize the training Lim and Ye [2017] propose to use the Hinge loss. Zhang et al. [2019] introduce SAGAN which uses a self-attention mechanism in the generator and discriminator to better capture long range dependencies. Combining those different techniques Brock et al. [2019] were able to train GANs on imagenet and produced high quality images up to 512x512 resolution. In parallel, Karras et al. [2019] proposed StyleGAN another architecture, able to generate high quality images of faces at a 1024x1024 resolution. Most recently, Sauer et al. [2022] improved the architecture and achieved state of the art results on generating images at 1024x1024 resolution. This impressive achievements also bring new ethical concerns, the use of such algorithms to spread disinformation, blackmail or discredit people has caught the attention of the media and coined the term deep fake, to refer to videos that have been manipulated using deep learning algorithms.

Most GANs objectives are derived from the dual formulation of the corresponding divergence they are trying to minimize. For example the family of f-divergence, admits the following dual formulation:

$$\mathrm{Div}(P_\theta, P_\mathcal{D}) = \sup_{D:\mathcal{X}\to\mathbb{R}} \mathbb{E}_{x\sim P_\theta}[D(x)] - \mathbb{E}_{x'\sim P_\mathcal{D}}[f^*(D(x'))] \tag{4.6}$$

where f^* is the convex conjugate of f.

This dual formulation is useful because it does not requires to know neither p_θ nor $p_\mathcal{D}$ to be computed, but instead we only need to be able to draw samples from p_θ and $p_\mathcal{D}$ to estimate it.

The Wasserstein-1 distance also admits a dual representation:

$$W_1(P_\theta, P_\mathcal{D}) = \sup_{\|D\|_L\leq 1} \mathbb{E}_{x\sim P_\theta}[D(x)] - \mathbb{E}_{x'\sim P_\mathcal{D}}[D(x')] \tag{4.7}$$

Following Huang et al. [2017a], we adopt a unifying view and propose to call this type of divergence, adversarial divergence:

$$\mathrm{Div}_{\mathrm{adv}}(P_\theta, P_\mathcal{D}) = \sup_{D\in\mathcal{D}} \mathbb{E}_{x\sim P_\theta, x'\sim P_\mathcal{D}}[\Delta(D(x), D(x'))] \tag{4.8}$$

where D is called the discriminator. The choice of family $\mathcal{D}$ and of δ will induce different divergence with different properties. For example choosing $\mathcal{D}$ to be the set of 1-Lipshitz function and choosing $\Delta(x, x') = x - x'$, is equivalent to the Wassertein-1 distance.

Finally we introduce the notion of parametric divergence, where the discriminator

D is parametric, and $\mathcal{D}$ is parametric class of functions:

$$\mathrm{Div}_\phi(P_\theta, P_\mathcal{D}) = \sup_{\phi\in\Phi} \mathbb{E}_{x\sim P_\theta, x'\sim P_\mathcal{D}}[\Delta(D_\phi(x), D_\phi(x'))] \tag{4.9}$$

In addition GANs use a parametric generator that generates samples $x \sim P_\theta$ through the following process:

$$x = g_\theta(\epsilon) \text{ with } \epsilon \sim P_\epsilon \tag{4.10}$$

where P_ϵ could be any distribution but in practice the normal or uniform distribution are often used.

By combining this parametric generator with the parametric divergence defined above, we can derive the min-max problem GANs are trying to solve:

$$\min_\theta \max_\phi \mathbb{E}_{\epsilon\sim P_\epsilon, x'\sim P_\mathcal{D}}[\Delta(D_\phi(g_\theta(\epsilon)), D_\phi(x'))] \tag{4.11}$$

Most GAN formulations can be viewed as an instance of this problem, with difference choice of Δ, Θ and Φ leading to different formulations with different properties and solutions which are not yet fully understood. For an overview and comparison about the properties of the different adversarial divergence refer to Huang et al. [2017a].

Several papers proposed to apply separate regularization to the generator and the discriminator Gulrajani et al. [2017], we can adopt a more general game formulation that can capture this regularization, where we define different objective functions for the generator and the discriminator:

$$\min_\theta \mathcal{L}_\theta(\theta, \phi) \quad \text{and} \quad \min_\phi \mathcal{L}_\phi(\theta, \phi) \tag{4.12}$$

where$\mathcal{L}_\theta(\theta, \phi)$ is the loss of the generator and $\mathcal{L}_\phi(\theta, \phi)$ is the loss of the discriminator.

The goal of GANs is to find the equilibrium of this game. Given that both the generator and discriminator are expressive enough this game admits a Nash-Equilibrium where the optimal samples generated by the generator are identical to the real samples.

This type of problem is a particular instance of a 2-player game, we will discuss multi-player games in Section 7.

5 Adversarial Learning

Along the great success and deployment of machine learning models in the real world, some concerns have been raised about the security of such system. One such concern is the ability to manipulate the prediction of a model through different type of attacks. In this section we will look at a particular type of attacks called Adversarial attacks.

Adversarial Attacks. Adversarial attacks is a type of attack that is able to disrupt the prediction of a model by adding a small amount of noise to the input of the model. Here we will take the point of view of the attacker and explain how someone with ill-intentions could construct such an attack. We will look at such problems in the context of classification.

As a reminder the classification problem is the following:

$$\min_\theta \mathbb{E}_{x,y\sim P_\mathcal{D}}[\mathcal{L}(y, f_\theta(x))] \tag{5.1}$$

where $f_\theta(x)$ is the prediction of the model for input x.

The goal of the attacker is to solve the following problem:

$$\max_\epsilon \mathcal{L}(y, f_\theta(x+\epsilon)) \quad \text{with} \quad ||\epsilon|| \leq r \tag{5.2}$$

We impose a constraint on the perturbation the attacker can produce, since the goal of the attacker is to remain undetected we constraint the attack to be in a ball of small radius ϵ.

Two types of attacks have been proposed. In black-box attacks, we assume the attacker can only query the output of the model but does not have access to the parameters of the model and thus the attacker cannot compute the gradient of the model. In white-box attacks, the attacker also has access to the parameters and the gradient of the model.

Several works proposed to use gradient based methods to craft such attacks. Goodfellow et al. [2015a] proposed the fast gradient sign method (FGSM) by taking the sign of the gradient of the loss function, and Madry et al. [2018] proposed to use projected gradient descent (PGD) to solve (5.2).

Adversarial training.

To prevent such attacks several defense strategies have been proposed. One popular solution is to formulate the problem as a robust optimization problem.

$$\min_{\theta} \mathbb{E}_{x,y\sim p_{\mathcal{D}}}[\max_{\epsilon} \mathcal{L}(y, f_\theta(x+\epsilon))] \tag{5.3}$$

To solve this problem Madry et al. [2018] proposed at each iteration to construct attacks using PGD and then train the model on both the original and perturbed examples. This method greatly improved the robustness of the model to such attacks.

6 Optimization

In previous sections we have looked at several machine learning problems, we have seen that a lot of those problems can be formulated as unconstrained minimization problem:

$$\min_{\theta} f(\theta) \tag{6.1}$$

In this section we will present some useful concept and algorithms from optimization to solve such problems. We first introduce some definitions and notions that are useful:

Convexity. A function f is said to be convex if it satisfies the following condition:

$$f(tx+(1-t)y) \leq tf(x) + (t-1)f(y) \tag{6.2}$$

If f is differentiable another equivalent condition is the following:

$$f(x) - f(y) \leq \nabla f(x)(x-y) \tag{6.3}$$

Strong-convexity. A slightly stronger condition for a function is for the function to be μ-strongly convex:

$$f(tx+(1-t)y) \leq tf(x) + (t-1)f(y) + \frac{\mu}{2}||x-y||^2 \tag{6.4}$$

Smoothness. Another useful notion is the notion of smoothness, a function is said to be L-smooth if its gradient is L-Lipschitz:

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y|| \tag{6.5}$$

If the function is twice differentiable we can characterize the spectrum of the Hessian of f, denoted by $\nabla^2 f$:

$$f \text{ is } \mu\text{-strongly convex} \quad \text{iff} \quad \nabla^2 f \succeq \mu \tag{6.6}$$

$$f \text{ is } L\text{-smooth} \quad \text{iff} \quad -L \preceq \nabla^2 f \preceq L \tag{6.7}$$

6.1 Gradient Descent

To solve (6.1), we can use iterative methods. A popular iterative method to solve this type of problems are gradient based methods. The most simple version is called Gradient Descent:

$$\theta_{k+1} = \theta_k - \gamma \nabla f(\theta_k) \tag{6.8}$$

where γ is called the step-size or the learning rate.

If we assume that f is μ-strongly convex and L-smooth, we can show that gradient descent converges linearly to the solution with $\gamma = \frac{1}{L}$, see Gower [2018] for a proof:

$$||x_k - x^*||^2 \leq \left(1 - \frac{1}{\kappa}\right)^k ||x_0 - x^*|| \tag{6.9}$$

Where $\kappa = \frac{L}{\mu}$ is called the condition number, this number provides an indication of the tightness on the bounds of the curvature of f, and thus of the difficulty to optimize f, larger κ will lead to slower convergence.

This rate of convergence can be improved by using accelerated methods, for example Nesterov [2013] proposed Nesterov's acceleration which is optimal for this type of problems. For an overview of accelerated methods see d'Aspremont et al. [2021]. Other methods have been proposed, some are specifically tailored to deep learning applications such as Adam [Kingma and Ba, 2015]. For further information, Ruder [2016] provides an overview of existing algorithms for training deep learning models.

6.2 Stochastic Gradient Descent

In ML, we are often interested in solving stochastic minimization problems of the form:

$$\min_{\theta} \mathbb{E}_{x\sim p_{\mathcal{D}}}[f(\theta;x)] \tag{6.10}$$

A particular case of this is when we have a finite dataset, and have the following finite sum problem:

$$\min_{\theta} \sum_{i=1}^{n} f_i(\theta) \tag{6.11}$$

In order to solve this type of problems we can use stochastic algorithms that use unbiased estimate of the gradients g_k such that $\mathbb{E}[g_k] = \mathbb{E}_{x\sim p_{\mathcal{D}}}[f(\theta_k;x)]$, by using unbiased estimate of the gradient with gradient descent we get the stochastic gradient descent (SGD) algorithm:

$$\theta_{k+1} = \theta_k - \gamma g_k \tag{6.12}$$

Contrarly to the deterministic case, SGD does not converge to the solution but instead only converges to a neighborhood of the solution when the function is μ-strongly convex and L-smooth. In order to converge to the exact solution, we need to use a decreasing step-size, however by doing so we lose the linear convergence guarantees. To fix this issue, variations have been proposed that use variance reduction techniques to get linear convergence to the solution. For example, Johnson and Zhang [2013] proposed SVRG a method that computes the full batch gradient every n iterations. Defazio et al. [2014] proposed SAGA another variance reduced method, that keeps the gradients of all the samples in memory.

6.3 Non-convex optimization

In practice the problems we are trying to solve are often non-convex and thus the assumptions we made to provide convergence guarantees do not hold anymore. However it has been observed, in the case of deep neural networks, that using gradient methods often perform really well and converge to very good solutions. Several works have tried to provide convergence guarantees for non-convex problems. [Gower et al., 2019] derived convergence guarantees for SGD for a class of non-convex problem which has a unique global minimum. Allen-Zhu et al. [2019] show that when a neural network is overparametrized, i.e. the number of parameters in the model is large compared to the number of training examples, SGD converges to a global minimum of the training objective.

Some good reference on optimization includes Boyd and Vandenberghe [2004], Bubeck et al. [2015], Hazan [2019].

7 Smooth multi-player games

In the previous section we looked at unconstrained minimization problem. However some problem such as GANs requires solving a different type of problems, where instead of minimizing a single objective, the goal is to find the Nash-Equilibrium of a game between two players with opposite objectives, where the solution of a player depends on the solution of the other player. This type of games are referred to as multi-player games.

In multi-player games we can have n different players, each with their own objective function and with their own set of parameters. We will denote the loss and parameter of player k by $\mathcal{L}_k$ and θ_k, respectively. We will also define a vector θ as the concatenation of the parameters of all the players. Each player is trying to minimize its own loss, we are thus trying to solve n problems simultaneously:

$$\begin{gathered}\min_{\theta_1} \mathcal{L}_1(\theta) \\ \vdots \\ \min_{\theta_n} \mathcal{L}_n(\theta)\end{gathered}$$

In the rest of this thesis, we will assume that all $\mathcal{L}_k$ are twice differentiable.

When the losses $\mathcal{L}_k$ are differentiable, another way to represent such a game is through its vector field, which is the concatenation of the loss of all the players:

$$F(\theta) = \begin{bmatrix} \nabla \mathcal{L}_1(\theta) \\ \vdots \\ \nabla \mathcal{L}_n(\theta) \end{bmatrix} \tag{7.1}$$

Solving this type of problems require finding equilibrium points. We define equilibrium points as stationary points of the vector field $F(\theta)$ where $F(\theta) = 0$. Several notion of equilibrium and optimality have been proposed in the literature. A first notion of optimality was introduced by Nash et al. [1950], are called Nash-Equilibria. A Nash-equlibrium is a point such that no player can improve its own loss by deviating from its current strategy given all the other players remain fixed. We can characterize such point by their first and second order condition:

$$\text{First order condition: } F(\theta) = 0$$
$$\text{Second order condition: } \nabla^2_{\theta_k} \mathcal{L}_k(\theta) \succeq 0 \quad \forall k \in \{1, \cdots, n\}$$

Another notion of optimality that comes from the dynamical system literature comes, is the notion of locally stable stationary point (LSSP). A point is locally stable, if for every possible small perturbations in a radius ϵ, the dynamic of the game $F(\theta)$ will converge back to that point. Such points can also be characterized using first and second order conditions:

$$\text{First order condition: } F(\theta) = 0$$
$$\text{Second order condition: } \Re(\lambda) \geq 0 \quad \forall \lambda \in Sp(\nabla F(\theta))$$

Variational Inequality Problem (VIP). In the previous section we considered the unconstrained case, however sometimes we might consider some problems where we add constraints on θ. For example, Arjovsky et al. [2017] propose to constraint the parameters of the discriminator to lie inside a l_∞ ball.

In that case we can formulate the problem of finding a stationary point of the game as a variational inequality problem :

$$\text{find } \theta^* \in \Theta \quad \text{such that} \quad F(\theta^*)^T(\theta - \theta^*) \geq 0, \quad \forall \theta \in \Theta \tag{7.2}$$

Solving this problem is equivalent to find stationary point of the vector field $F(\theta)$.

We can generalize the notion of convexity, by defining motonicity. F is said to be monotone iff:

$$(F(\theta) - F(\theta'))^T(\theta - \theta') \geq 0 \quad \forall \theta, \theta' \in \Theta \tag{7.3}$$

For further details on VIP, see Harker and Pang [1990], Facchinei and Pang.

Algorithms. Several algorithms have been proposed to solve VIP. In particular, gradient descent is not guaranteed to converge to the solution of the game when F is monotone. To adress this issue, [Korpelevich, 1976] proposed extragradient, an algorithm that uses an extrapolation step before updating the parameters. More recently [Mescheder et al., 2017] proposed two new algorithms hamiltonian gradient descent (HGD) and consensus optimization to solve games by minimizing the norm of F. [Loizou et al., 2020] derived convergence guarantees for SHGD on a special class of nonconvex-nonconcave games. Finally, Gidel et al. [2019b] proposed to fix gradient descent by adding a negative momentum term.

3 Prologue to the First Contribution

1 Article Details

A Variational Inequality Perspective on Generative Adversarial Networks. *Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent and Simon Lacoste-Julien.* This paper was published at ICLR 2019 [Gidel et al., 2019a].

2 Contributions of the authors

This contribution resulted from the interaction between Hugo Berard interested in GANs and Gauthier Gidel interested in min-max optimization. By merging their expertise they were able to bring tools and results from the optimization literature to the GAN community. Hugo Berard contributed his knowledge and experience about training GANs, while Gauthier Gidel contributed his knowledge of optimization and variatonal inequality. The idea of ExtraAdam emerged from combining those two expertises. They both contributed to the writing of the paper, with Hugo Berard focusing more on GANs and on the experimental results and Gauthier Gidel focusing more on VIP and on the theoretical results. Gaëtan Vignoud came up with the idea for extragradient from the past and proof-read the proofs of the paper. Pascal Vincent and Simon Lacoste-Julien supervised the project, Simon Lacoste-Julien also orignially came up with the idea of making a connection between the variational inequality literature and GANs.

3 Context and Impact

At the time of writing this article GANs started to become hugely popular and the first large-scale results on CIFAR-10 and CelebA were published. However the quality of the results were still far from perfect and generative models could only generate relatively low dimensional images. Furthermore GANs were notoriously hard to train requiring a lot of fine-tuning of the hyper-parameters, and requiring a lot of tricks [Salimans et al., 2016] to make the training more stable. Most of the theory about GANs adopted a divergence perspective. For example, Arjovsky et al. [2017] proposed the Wasserstein distance as a solution to the limitations of the KL divergence. Based on those theoretical results people started using different regularization techniques to stabilize training [Gulrajani et al., 2017, Miyato et al., 2018]. One of the first article, to look at the instability in GANs from an optimization perspective was Mescheder et al. [2017], they were the first to show that instabilities during training came from the adversarial nature of the game that leads to the Jacobian of the game having imaginary eigenvalues that make gradient descent-ascent diverge. Starting from this observation and by tapping in the optimization literature, we tried in this work to bridge the gap between the optimization literature and GANs. To do so we cast GANs as an instance of the variational inequality problem. Leveraging the vast literature on variational inequality, we proposed to use extragradient and combined it to Adam, a popular method used to train GANs, resulting in a new algorithm that we call ExtraAdam. This work also highlighted some of the limitations of current theoretical results. Since then several works have tried to extend the theory to provide better algorithms and guarantees for stochastic and non-monotone games. A non-exhaustive list of papers that extend this work includes, Lin et al. [2020] who provides convergence guarantees for GDA on nonconvex-concave games. Yang et al. [2020] that derive convergence guarantees for SGDA on a class of nonconvex-nonconcave games. Chavdarova et al. [2019] proposes a variance-reduced version of extragradient to train GANs. Gorbunov et al. [2022] provide a general analysis of extragradient for quasi-strongly-monotone VIP. Some observations made in the paper such as for example the observation that averaging can help improve performance are now widely used in the GAN literature [Brock et al., 2019].

Limitations and Remarks. We would like to mention some limitations of this work. First, we could not derive an algorithm with last-iterate convergence guarantees for monotone games in the stochastic setting. Extragradient by itself was not able to improve over Adam, instead we proposed to combine extragradient with Adam resulting in a new algorithm called ExtraAdam, this improved the results over Adam and training was more stable. However it still required careful tuning of the hyper-parameters in particular the different step-sizes for the generator and the discriminator. It is still not clear why Adam is necessary to achieve good performance, Jelassi et al. [2021] proposed an explanation showing that the main advantage of Adam comes from the normalization of the gradients. Choosing different step-size for both players seemed to have a major impact on performance, we were not able to provide any theoretical reason for this observation.

4 A Variational Inequality Perspective on Generative Adversarial Networks

Abstract

Generative adversarial networks (GANs) form a generative modeling approach known for producing appealing samples, but they are notably difficult to train. One common way to tackle this issue has been to propose new formulations of the GAN objective. Yet, surprisingly few studies have looked at optimization methods designed for this adversarial training. In this work, we cast GAN optimization problems in the general variational inequality framework. Tapping into the mathematical programming literature, we counter some common misconceptions about the difficulties of saddle point optimization and propose to extend techniques designed for variational inequalities to the training of GANs. We apply *averaging*, *extrapolation* and a novel computationally cheaper variant that we call *extrapolation from the past* to the stochastic gradient method (SGD) and Adam.

1 Introduction

Generative adversarial networks (GANs) [Goodfellow et al., 2014] form a generative modeling approach known for producing realistic natural images [Karras et al., 2018] as well as high quality super-resolution [Ledig et al., 2017] and style transfer [Zhu et al., 2017]. Nevertheless, GANs are also known to be difficult to train, often displaying an unstable behavior [Goodfellow, 2016]. Much recent work has tried to tackle these training difficulties, usually by proposing new formulations of the GAN objective [Nowozin et al., 2016, Arjovsky et al., 2017]. Each of these formulations can be understood as a two-player game, in the sense of game theory [Neumann and Morgenstern, 1944], and can be addressed as a variational inequality problem (VIP) [Harker and Pang, 1990], a framework that encompasses traditional saddle point optimization algorithms [Korpelevich, 1976].

Solving such GAN games is traditionally approached by running variants of stochastic gradient descent (SGD) initially developed for optimizing supervised neural network objectives. Yet it is known that for some games [Goodfellow, 2016, §8.2] SGD exhibits oscillatory behavior and fails to converge. This oscillatory behavior, which does not arise from stochasticity, highlights a fundamental problem: while a direct application of basic gradient descent is an appropriate method for regular minimization problems, it is *not* a sound optimization algorithm for the kind of two-player games of GANs. This constitutes a fundamental issue for GAN training, and calls for the use of more principled methods with more reassuring convergence guarantees.

Contributions.. We point out that multi-player games can be cast as *variational inequality problems* and consequently the same applies to any GAN formulation posed as a minimax or non-zero-sum game. We present two techniques from this literature, namely *averaging* and *extrapolation*, widely used to solve variational inequality problems (VIP) but which have not been explored in the context of GANs before.[1]

We extend standard GAN training methods such as SGD or Adam into variants that incorporate these techniques (Alg. 3, 4 are new). We also explain that the oscillations of basic SGD for GAN training previously noticed [Goodfellow, 2016] can be explained by standard variational inequality optimization results and we illustrate how *averaging* and *extrapolation* can fix this issue.

We introduce a new technique, called *extrapolation from the past*, that only requires one gradient computation per iteration compared to *extrapolation* which requires to compute the gradient twice. We *prove its convergence* in the stochastic variational inequality setting, i.e. when applied to SGD.

Finally, we test these techniques in the context of standard GAN training. We observe a 4%-6% improvement on the inception score [Salimans et al., 2016] of WGAN [Arjovsky et al., 2017] and WGAN-GP [Gulrajani et al., 2017] on the CIFAR-10 dataset.

Outline.. §2 presents the background on GAN and optimization, and shows how to cast this optimization as a VIP. §3 presents standard techniques to optimize variational inequalities in a batch setting as well as our new one, *extrapolation from the past*. §4 considers these methods in the *stochastic* setting, yielding three corresponding variants of SGD, and provides their respective convergence rates. §5 develops how to combine these techniques with already existing algorithms. §6 discusses the related work and §7 presents experimental results.

[1]Independent works [Mertikopoulos et al., 2019] and [Yazıcı et al., 2019] respectively explored extrapolation and averaging in the context of GANs. More details in related work section §6.

2 GAN optimization as a variational inequality problem

2.1 GAN formulations

The purpose of generative modeling is to generate samples from a distribution $q_{\boldsymbol{\theta}}$ that matches best the true distribution p of the data. The generative adversarial network training strategy can be understood as a *game* between two players called *generator* and *discriminator*. The former produces a sample that the latter has to classify between real or fake data. The final goal is to build a generator able to produce sufficiently realistic samples to fool the discriminator.

In the original GAN paper [Goodfellow et al., 2014], the GAN objective is formulated as a *zero-sum game* where the cost function of the discriminator $D_{\boldsymbol{\varphi}}$ is given by the negative log-likelihood of the binary classification task between real or fake data generated from $q_{\boldsymbol{\theta}}$ by the generator,

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\varphi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \quad \text{where} \quad \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) := -\mathop{\mathbb{E}}_{\mathbf{x}\sim p}[\log D_{\boldsymbol{\varphi}}(\mathbf{x})] - \mathop{\mathbb{E}}_{\mathbf{x}'\sim q_{\boldsymbol{\theta}}}[\log(1-D_{\boldsymbol{\varphi}}(\mathbf{x}'))] \,. \tag{2.1}$$

However Goodfellow et al. [2014] recommends to use in practice a second formulation, called *non-saturating GAN*. This formulation is a *non-zero-sum game* where the aim is to jointly minimize:

$$\mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\theta}, \boldsymbol{\varphi}) := -\mathop{\mathbb{E}}_{\mathbf{x}'\sim q_{\boldsymbol{\theta}}} \log D_{\boldsymbol{\varphi}}(\mathbf{x}') \;\text{ and }\; \mathcal{L}^{(\boldsymbol{\varphi})}(\boldsymbol{\theta}, \boldsymbol{\varphi}) := -\mathop{\mathbb{E}}_{\mathbf{x}\sim p} \log D_{\boldsymbol{\varphi}}(\mathbf{x}) - \mathop{\mathbb{E}}_{\mathbf{x}'\sim q_{\boldsymbol{\theta}}} \log(1-D_{\boldsymbol{\varphi}}(\mathbf{x}')) \,. \tag{2.2}$$

The dynamics of this formulation have the same *stationary points* as the zero-sum one (2.1) but are claimed to provide "much stronger gradients early in learning" [Goodfellow et al., 2014] .

2.2 Equilibrium

The minimax formulation (2.1) is theoretically convenient because a large literature on games studies this problem and provides guarantees on the existence of equilibria. Nevertheless, practical considerations lead the GAN literature to consider a different objective for each player as formulated in (2.2). In that case, the *two-player game problem* [Neumann and Morgenstern, 1944] consists in finding the following *Nash equilibrium*:

$$\boldsymbol{\theta}^* \in \mathop{\arg\min}_{\boldsymbol{\theta}\in\Theta} \mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\theta}, \boldsymbol{\varphi}^*) \quad \text{and} \quad \boldsymbol{\varphi}^* \in \mathop{\arg\min}_{\boldsymbol{\varphi}\in\Phi} \mathcal{L}^{(\boldsymbol{\varphi})}(\boldsymbol{\theta}^*, \boldsymbol{\varphi}) \,. \tag{2.3}$$

Only when $\mathcal{L}^{(\boldsymbol{\theta})} = -\mathcal{L}^{(\boldsymbol{\varphi})}$ is the game called a *zero-sum game* and (2.3) can be formulated as a minimax problem. One important point to notice is that the two optimization problems in (2.3) are *coupled* and have to be considered *jointly* from an optimization point of view.

Standard GAN objectives are non-convex (i.e. each cost function is non-convex), and thus such (pure) equilibria may not exist. As far as we know, not much is known about the existence of these equilibria for non-convex losses (see Heusel et al. [2017] and references therein for some results). In our theoretical analysis in §4, our assumptions (monotonicity (4.1) of the operator and convexity of the constraints set) imply the existence of an equilibrium.

In this paper, we focus on ways to optimize these games, assuming that an equilibrium exists. As is often standard in non-convex optimization, we also focus on finding points satisfying the necessary *stationary conditions*. As we mentioned previously, one difficulty that emerges in the optimization of such games is that the two different cost functions of (2.3) have to be minimized jointly in $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$. Fortunately, the optimization literature has for a long time studied so-called *variational inequality problems*, which generalize the stationary conditions for two-player game problems.

2.3 Variational inequality problem formulation

We first consider the local necessary conditions that characterize the solution of the *smooth* two-player game (2.3), defining *stationary points*, which will motivate the definition of a variational inequality. In the unconstrained setting, a *stationary point* is a couple $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$ with zero gradient:

$$\|\nabla_{\boldsymbol{\theta}} \mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)\| = \|\nabla_{\boldsymbol{\varphi}} \mathcal{L}^{(\boldsymbol{\varphi})}(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)\| = 0\,. \tag{2.4}$$

When constraints are present,[2] a *stationary point* $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$ is such that the directional derivative of each cost function is non-negative in any feasible direction (i.e. there is no feasible descent direction):

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}^*) \geq 0 \quad \text{and} \quad \nabla_{\boldsymbol{\varphi}} \mathcal{L}^{(\boldsymbol{\varphi})}(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)^\top (\boldsymbol{\varphi} - \boldsymbol{\varphi}^*) \geq 0\,, \quad \forall\,(\boldsymbol{\theta}, \boldsymbol{\varphi}) \in \Theta \times \Phi. \tag{2.5}$$

Defining $\boldsymbol{\omega} := (\boldsymbol{\theta}, \boldsymbol{\varphi})$, $\boldsymbol{\omega}^* := (\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$, $\Omega := \Theta \times \Phi$, Eq. (2.5) can be compactly formulated as:

$$F(\boldsymbol{\omega}^*)^\top (\boldsymbol{\omega} - \boldsymbol{\omega}^*) \geq 0\,, \quad \forall \boldsymbol{\omega} \in \Omega \quad \text{where} \quad F(\boldsymbol{\omega}) := \left[\nabla_{\boldsymbol{\theta}} \mathcal{L}^{(\boldsymbol{\theta})}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \quad \nabla_{\boldsymbol{\varphi}} \mathcal{L}^{(\boldsymbol{\varphi})}(\boldsymbol{\theta}, \boldsymbol{\varphi})\right]^\top. \tag{2.6}$$

[2] An example of constraint for GANs is to clip the parameters of the discriminator [Arjovsky et al., 2017].

These stationary conditions can be generalized to any continuous vector field: let $\Omega \subset \mathbb{R}^d$ and $F : \Omega \to \mathbb{R}^d$ be a continuous mapping. The *variational inequality problem* [Harker and Pang, 1990] (depending on F and Ω) is:

$$\text{find } \boldsymbol{\omega}^* \in \Omega \quad \text{such that} \quad F(\boldsymbol{\omega}^*)^\top(\boldsymbol{\omega} - \boldsymbol{\omega}^*) \geq 0\,, \;\; \forall \boldsymbol{\omega} \in \Omega\,. \tag{VIP}$$

We call *optimal set* the set Ω^* of $\boldsymbol{\omega} \in \Omega$ verifying (VIP). The intuition behind it is that any $\boldsymbol{\omega}^* \in \Omega^*$ is a *fixed point* of the *constrained* dynamic of F (constrained to Ω).

We have thus showed that both saddle point optimization and non-zero sum game optimization, which encompasses the large majority of GAN variants proposed in the literature, can be cast as Variational Inequality Problems. In the following section, we turn to suitable optimization techniques for such problems.

3 Optimization of Variational Inequalities (batch setting)

Let us begin by looking at techniques that were developed in the optimization literature to solve (VIP). We present the intuitions behind them as well as their performance on a simple bilinear problem (see Fig. 4.1). Our goal here is to provide mathematical insights into the techniques of *averaging* (§3.1) and *extrapolation* (§3.2), to inspire their application to extending other optimization algorithm. We then propose a novel variant of the extrapolation technique in §3.3 *extrapolation from the past*. We here treat the batch setting, i.e. considering that the operator $F(\boldsymbol{\omega})$ as defined in Eq. 2.6 yields an exact full gradient. We will present extensions of these techniques to the stochastic setting later in §4.

The two standard methods studied in the VIP literature are the *gradient method* [Bruck, 1977] and the *extragradient method* [Korpelevich, 1976]. The iterates of the basic gradient method are given by $\boldsymbol{\omega}_{t+1} = P_\Omega[\boldsymbol{\omega}_t - \eta F(\boldsymbol{\omega}_t)]$ where $P_\Omega[\cdot]$ is the *projection onto the constraints set* (if constraints are present) associated to (VIP). These iterates are known to converge linearly under an additional assumption on the operator[3] [Chen and Rockafellar, 1997], but oscillate for a bilinear operator as shown in Fig. 4.1. On the other hand, the *uniform average* of these iterates converge for any bounded monotone operator with a $O(1/\sqrt{t})$ rate [Nedić and Ozdaglar, 2009], motivating the presentation of *averaging* in §3.1. By contrast, the *extragradient method* (extrapolated gradient) does not require any

[3]Strong monotonicity, a generalization of strong convexity. See §1.

averaging to converge for monotone operators (in the batch setting), and can even converge at the faster $O(1/t)$ rate [Nesterov, 2007]. The idea of this method is to compute a lookahead step (see intuition on *extrapolation* in §3.2) in order to compute a more stable direction to follow.

3.1 Averaging

More generally, we consider a *weighted averaging* scheme with weights $\rho_t \geq 0$. This *weighted averaging* scheme have been proposed for the first time for (batch) VIP by Bruck [1977],

$$\bar{\boldsymbol{\omega}}_T := \frac{\sum_{t=0}^{T-1} \rho_t \boldsymbol{\omega}_t}{S_T}, \quad S_T := \sum_{t=0}^{T-1} \rho_t \,. \tag{3.1}$$

Averaging schemes can be efficiently implemented in an online fashion noticing that,

$$\bar{\boldsymbol{\omega}}_t = (1-\tilde{\rho}_t)\bar{\boldsymbol{\omega}}_{t-1} + \tilde{\rho}_t \boldsymbol{\omega}_t \quad \text{where} \quad 0 \leq \tilde{\rho}_t \leq 1 \,. \tag{3.2}$$

For instance, setting $\tilde{\rho}_t = \frac{1}{t}$ provides *uniform averaging* ($\rho_t = 1$) and $\tilde{\rho}_t = 1 - \beta < 1$ provides *geometric averaging* also known as *exponential moving averaging* ($\rho_t = \beta^t$). Averaging is experimentally compared with the other techniques presented in this section in Fig. 4.1.

In order to illustrate how averaging tackles the oscillatory behavior in game optimization, we consider a toy example where the discriminator and the generator are linear: $D_{\boldsymbol{\varphi}}(\mathbf{x}) = \boldsymbol{\varphi}^T \mathbf{x}$ and $G_{\boldsymbol{\theta}}(\mathbf{z}) = \boldsymbol{\theta}\mathbf{z}$ (implicitly defining $q_{\boldsymbol{\theta}}$). By replacing these expressions in the WGAN objective,[4] we get the following bilinear objective:

$$\min_{\boldsymbol{\theta} \in \Theta} \max_{\boldsymbol{\varphi} \in \Phi, ||\boldsymbol{\varphi}|| \leq 1} \boldsymbol{\varphi}^T \mathbb{E}[\mathbf{x}] - \boldsymbol{\varphi}^T \boldsymbol{\theta} \mathbb{E}[\mathbf{z}] \,. \tag{3.3}$$

A similar task was presented by Nagarajan and Kolter [2017] where they consider a quadratic discriminator instead of a linear one, and show that gradient descent is not necessarily asymptotically stable. The bilinear objective has been extensively used [Goodfellow, 2016, Mescheder et al., 2018, Yadav et al., 2018] to highlight the difficulties of gradient descent for saddle point optimization. Yet, ways to cope with this issue have been proposed decades ago in the context of mathematical programming. Simplifying further by setting the dimension to 1 and centering the equilibrium to the origin, Eq. 3.3 becomes:

$$\min_{\theta \in \mathbb{R}} \max_{\phi \in \mathbb{R}} \; \theta \cdot \phi \qquad \text{and} \qquad (\theta^*, \phi^*) = (0,0) \,. \tag{3.4}$$

[4]Wasserstein GAN (WGAN) proposed by Arjovsky et al. [2017] boils down to the following minimax formulation: $\min_{\boldsymbol{\theta} \in \Theta} \max_{\boldsymbol{\varphi} \in \Phi, ||D_{\boldsymbol{\varphi}}||_L \leq 1} \mathbb{E}_{\mathbf{x} \sim p}[D_{\boldsymbol{\varphi}}(\mathbf{x})] - \mathbb{E}_{\mathbf{x}' \sim q_{\boldsymbol{\theta}}}[D_{\boldsymbol{\varphi}}(\mathbf{x}')]$.

The operator associated with this minimax game is $F(\theta,\phi)=(\phi,-\theta)$. There are several ways to compute the discrete updates of this dynamics. The two most common ones are the *simultaneous* and the *alternated* gradient update rules,

$$\text{Simultaneous: } \begin{cases}\theta_{t+1}=\theta_t-\eta\phi_t\\ \phi_{t+1}=\phi_t+\eta\theta_t\end{cases}, \qquad \text{Alternated: } \begin{cases}\theta_{t+1}=\theta_t-\eta\phi_t\\ \phi_{t+1}=\phi_t+\eta\theta_{t+1}\end{cases}. \tag{3.5}$$

Interestingly, these two choices give rise to have a completely different behavior. The norm of the *simultaneous* updates diverges geometrically whereas the alternated iterates are bounded but do not converge to the equilibrium. As a consequence, their respective uniform average has a different behavior, as highlighted in the following proposition (more details and proof in §2.1):

Proposition 1. *The* simultaneous *iterates diverge geometrically and the* alternated *iterates defined in (3.5) are bounded but do not converge to 0 as*

$$\text{Simultaneous: } \theta_{t+1}^2+\phi_{t+1}^2=(1+\eta^2)(\theta_t^2+\phi_t^2)\,, \qquad \text{Alternated: } \theta_t^2+\phi_t^2=\Theta(\theta_0^2+\phi_0^2) \tag{3.6}$$

where $u_t=\Theta(v_t)\Leftrightarrow \exists\alpha,\beta>0:\alpha v_t\le u_t\le\beta v_t$.

The uniform average $(\bar\theta_t,\bar\phi_t):=\frac{1}{t}\sum_{s=0}^{t-1}(\theta_s,\phi_s)$ *of the* simultaneous *updates (resp. the* alternated updates*) diverges (resp. converges to 0) as,*

$$\text{Simultaneous: } \bar\theta_t^2+\bar\phi_t^2=\Theta\left(\frac{\theta_0^2+\phi_0^2}{\eta^2t^2}(1+\eta^2)^t\right), \quad \text{Alternated: } \bar\theta_t^2+\bar\phi_t^2=\Theta\left(\frac{\theta_0^2+\phi_0^2}{\eta^2t^2}\right). \tag{3.7}$$

This sublinear convergence result, proved in §2, underlines the benefits of averaging when the sequence of iterates is bounded (i.e. for *alternated* update rule). When the sequence of iterates is not bounded (i.e. for *simultaneous* updates) averaging fails to ensure convergence. This theorem also shows how *alternated* updates may have better convergence properties than *simultaneous* updates.

3.2 Extrapolation

Another technique used in the variational inequality literature to prevent oscillations is *extrapolation*. This concept is anterior to the extragradient method since Korpelevich [1976] mentions that the idea of *extrapolated* "prices" to give "stability" had been already formulated by Polyak [1963, Chap. II]. The idea behind this technique is to compute the gradient at an (extrapolated) point different from the current point from which the update is performed, stabilizing the dynamics:

$$\text{Compute extrapolated point: } \boldsymbol{\omega}_{t+\frac{1}{2}}=P_\Omega[\boldsymbol{\omega}_t-\eta F(\boldsymbol{\omega}_t)]\,, \tag{3.8}$$

$$\text{Perform update step: } \boldsymbol{\omega}_{t+1}=P_\Omega[\boldsymbol{\omega}_t-\eta F(\boldsymbol{\omega}_{t+\frac{1}{2}})]\,. \tag{3.9}$$

Note that, even in the *unconstrained case*, this method is intrinsically different from Nesterov's momentum[5] [Nesterov, 1983, Eq. 2.2.9] because of this lookahead step for the gradient computation:

$$\text{Nesterov's method:} \quad \boldsymbol{\omega}_{t+\frac{1}{2}} = \boldsymbol{\omega}_t - \eta F(\boldsymbol{\omega}_t)\,, \qquad \boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_{t+\frac{1}{2}} + \beta(\boldsymbol{\omega}_{t+\frac{1}{2}} - \boldsymbol{\omega}_t)\,. \tag{3.10}$$

Nesterov's method does not converge when trying to optimize (3.4). One intuition explaining why *extrapolation* provides better convergence properties than the standard gradient method comes from Euler's method framework (see for instance [Atkinson, 2003] for more details on that topic). Actually, if we consider a first order approximation of $\boldsymbol{\omega}_{t+\frac{1}{2}}$, we have $\boldsymbol{\omega}_{t+\frac{1}{2}} \approx \boldsymbol{\omega}_{t+1} + o(\eta)$ and consequently, the update step (3.9) is close to an *implicit method* step:

$$\text{Implicit step:} \quad \boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta F(\boldsymbol{\omega}_{t+1})\,. \tag{3.11}$$

In the literature on Euler's method, *implicit methods* are known to be more stable and to benefit from better convergence properties [Atkinson, 2003] than *explicit methods*. They are not often used in practice though since they require to solve a potentially non-linear system at each iteration.

Taking back the simplified WGAN toy example (3.4) from §3.1 we get the following update rules,

$$\text{Implicit:} \begin{cases} \theta_{t+1} = \theta_t - \eta\phi_{t+1} \\ \phi_{t+1} = \phi_t + \eta\theta_{t+1} \end{cases}, \qquad \text{Extrapolation:} \begin{cases} \theta_{t+1} = \theta_t - \eta(\phi_t + \eta\theta_t) \\ \phi_{t+1} = \phi_t + \eta(\theta_t - \eta\phi_t) \end{cases}. \tag{3.12}$$

In the following proposition, we will see that the respective convergence rates of the *implicit method* and *extrapolation* are highly similar. Keeping in mind that the latter has the major advantage of being more practical, this proposition clearly underlines the benefits of *extrapolation* (more details and proof in §2.2),

Proposition 2. *The squared norm of the iterates $N_t := \theta_t^2 + \phi_t^2$, where the update rule of θ_t and ϕ_t are defined in (3.12), decreases geometrically for any $\eta < 1$ as,*

$$\text{Implicit:} \;\; N_{t+1} = \Big(1-\eta^2+\eta^4+\mathcal{O}(\eta^6)\Big)N_t\,, \qquad \text{Extrapolation:} \;\; N_{t+1} = (1-\eta^2+\eta^4)N_t\,. \tag{3.13}$$

3.3 Extrapolation from the past

One issue with extrapolation is that the algorithm "wastes" a gradient (3.8). Indeed we need to compute the gradient at two different positions for every single update

[5] Sutskever [2013, §7.2] showed the equivalence between "standard momentum" and Nesterov's formulation.

of the parameters. [Popov, 1980] proposed a similar technique that only requires a single gradient computation per update. The idea is to store and re-use the extrapolated gradient for the extrapolation:

$$\text{Extrapolation from the past:}\quad \boldsymbol{\omega}_{t+\frac{1}{2}} = P_\Omega[\boldsymbol{\omega}_t - \eta F(\boldsymbol{\omega}_{t-\frac{1}{2}})] \tag{3.14}$$

$$\text{Perform update step:}\quad \boldsymbol{\omega}_{t+1} = P_\Omega[\boldsymbol{\omega}_t - \eta F(\boldsymbol{\omega}_{t+\frac{1}{2}})] \tag{3.15}$$
$$\text{and store: } F(\boldsymbol{\omega}_{t+\frac{1}{2}})\,.$$

A similar update scheme was proposed by Chiang et al. [2012, Alg. 1] in the context of online convex optimization and generalized by Rakhlin and Sridharan [2013] for general online learning. Without projection, (3.14) and (3.15) reduce to the optimistic mirror descent described by Daskalakis et al. [2018]:

$$\text{Optimistic mirror descent (OMD):}\quad \boldsymbol{\omega}_{t+\frac{1}{2}} = \boldsymbol{\omega}_{t-\frac{1}{2}} - 2\eta F(\boldsymbol{\omega}_{t-\frac{1}{2}}) + \eta F(\boldsymbol{\omega}_{t-\frac{3}{2}}) \tag{3.16}$$

OMD was proposed with similar motivation as ours, namely tackling oscillations due to the game formulation in GAN training, but with an online learning perspective. Using the VIP point of view, we are able to prove a linear convergence rate for *extrapolation from the past* (see details and proof of Theorem 1 in §2.3). We also provide results on the averaged iterate for a stochastic version in §4. In comparison to the convergence results from Daskalakis et al. [2018] that hold for a bilinear objective, we provide a faster convergence rate (linear vs sublinear) on the last iterate for a general (strongly monotone) operator F and any projection on a convex Ω. One thing to notice is that the operator of a bilinear objective is *not* strongly monotone, but in that case one can use the standard extrapolation method (3.8) which converges linearly for an unconstrained bilinear game [Tseng, 1995, Cor. 3.3].

Theorem 1 (Linear convergence of *extrapolation from the past*)**.** *If F is μ-strongly monotone (see Appendix A §1 for the definition of strong monotonicity) and L-Lipschitz, then the updates* (3.14) *and* (3.15) *with $\eta = \frac{1}{4L}$ provide linearly converging iterates,*

$$\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2 \leq \left(1 - \frac{\mu}{4L}\right)^t \|\boldsymbol{\omega}_0 - \boldsymbol{\omega}^*\|_2^2\,, \quad \forall t \geq 0\,. \tag{3.17}$$

4 Optimization of VIP with stochastic gradients

In this section, we consider extensions of the techniques presented in section §3 for solving (VIP), to the context of a *stochastic* operator. In this case, at each time step we no longer have access to the exact gradient $F(\boldsymbol{\omega})$ but to an unbiased *stochastic* estimate of it $F(\boldsymbol{\omega}, \xi)$ where $\xi \sim P$ and $F(\boldsymbol{\omega}) := \mathbb{E}_{\xi\sim P}[F(\boldsymbol{\omega}, \xi)]$. This is

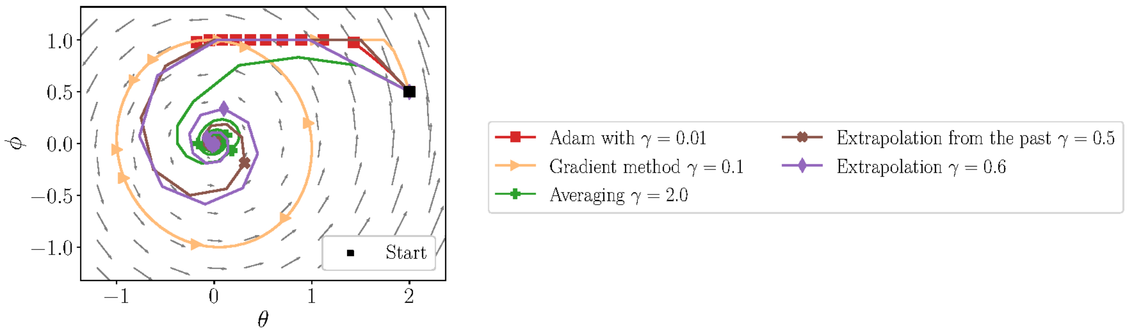


Figure 4.1: Comparison of the basic gradient method (as well as Adam) with the techniques presented in §3 on the optimization of (3.3). Only the algorithms advocated in this paper (Averaging, Extrapolation and Extrapolation from the past) converge quickly to the solution. Each marker represents 20 iterations. We compare these algorithms on a non-convex objective in §7.1.

Algorithm 1 AvgSGD
```
Let $\boldsymbol{\omega}_0 \in \Omega$
for $t = 0 \dots T-1$ do
    $\xi_t \sim P$                          (minibatch)
    $\boldsymbol{d}_t \leftarrow F(\boldsymbol{\omega}_t, \xi_t)$
    $\boldsymbol{\omega}_{t+1} \leftarrow P_\Omega[\boldsymbol{\omega}_t - \eta_t \boldsymbol{d}_t]$
end for
Return $\bar{\boldsymbol{\omega}}_T \leftarrow \frac{\sum_{t=0}^{T-1} \eta_t \boldsymbol{\omega}_t}{\sum_{t=0}^{T-1} \eta_t}$
```

Algorithm 2 AvgExtraSGD
```
for $t = 0 \dots T-1$ do
    $\xi_t, \xi'_t \sim P$                  (minibatches)
    $\boldsymbol{d}_t \leftarrow F(\boldsymbol{\omega}_t, \xi_t)$
    $\boldsymbol{\omega}'_t \leftarrow P_\Omega[\boldsymbol{\omega}_t - \eta_t \boldsymbol{d}_t]$
    $\boldsymbol{d}'_t \leftarrow F(\boldsymbol{\omega}'_t, \xi'_t)$
    $\boldsymbol{\omega}_{t+1} \leftarrow P_\Omega[\boldsymbol{\omega}_t - \eta_t \boldsymbol{d}'_t]$
end for
Return $\bar{\boldsymbol{\omega}}_T \leftarrow \frac{\sum_{t=0}^{T-1} \eta_t \boldsymbol{\omega}'_t}{\sum_{t=0}^{T-1} \eta_t}$
```

Algorithm 3 AvgPastExtraSGD
```
Let $\boldsymbol{\omega}_0 \in \Omega$
for $t = 0 \dots T-1$ do
    $\xi_t \sim P$                          (minibatch)
    $\boldsymbol{\omega}'_t \leftarrow P_\Omega[\boldsymbol{\omega}_t - \eta_t \boldsymbol{d}_{t-1}]$
    $\boldsymbol{d}_t \leftarrow F(\boldsymbol{\omega}'_t, \xi_t)$
    $\boldsymbol{\omega}_{t+1} \leftarrow P_\Omega[\boldsymbol{\omega}_t - \eta_t \boldsymbol{d}_t]$
end for
Return $\bar{\boldsymbol{\omega}}_T \leftarrow \frac{\sum_{t=0}^{T-1} \eta_t \boldsymbol{\omega}'_t}{\sum_{t=0}^{T-1} \eta_t}$
```

Figure 4.2: Three variants of SGD using the techniques introduced in §3.

motivated from the GAN formulation where we only have access to a finite sample estimate of the expected gradient, computed on a mini-batch. For GANs, ξ is thus a mini-batch of points coming from the true data distribution p and the generator distribution $q_{\boldsymbol{\theta}}$.

For our analysis, we require at least one of the two following assumptions on the stochastic operator:

Assumption 1. *Bounded variance by σ^2:* $\mathbb{E}_\xi[\|F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}, \xi)\|^2] \leq \sigma^2\,, \;\; \forall \boldsymbol{\omega} \in \Omega\,.$

Assumption 2. *Bounded expected squared norm by M^2:*

$$\mathbb{E}_\xi[\|F(\boldsymbol{\omega}, \xi)\|^2] \leq M^2, \; \forall \boldsymbol{\omega} \in \Omega.$$

Assump. 1 is standard in stochastic variational analysis, while Assump. 2 is a stronger assumption sometimes made in stochastic convex optimization. To illustrate how strong Assump. 2 is, note that it does not hold for an unconstrained bilinear objective like in our example 3.4 in §3. It is thus mainly reasonable for bounded constraint sets. Note that in practice we have $\sigma \ll M$.

We now present and analyse three algorithms, variants of SGD that are appropriate to solve (VIP). The first one Alg. 1 (AvgSGD) is the stochastic extension of the gradient method for solving (VIP); Alg. 2 (AvgExtraSGD) uses *extrapolation* and Alg. 3 (AvgPastExtraSGD) uses *extrapolation from the past.* A fourth variant (Alg.5) is proposed in §4. These three algorithms return an *average* of the iterates. The proofs of the theorems presented in this section are in §6.

To handle constraints such as parameter clipping [Arjovsky et al., 2017], we present a *projected* version of theses algorithms, where $P_\Omega[\boldsymbol{\omega}']$ denotes the projection of $\boldsymbol{\omega}'$ onto Ω (see §1). Note that when $\Omega = \mathbb{R}^d$, the projection is the identity mapping (unconstrained setting). In order to prove the convergence of these three algorithms we will assume that F is monotone:

$$(F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}'))^\top(\boldsymbol{\omega} - \boldsymbol{\omega}') \geq 0 \quad \forall\, \boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega\,. \tag{4.1}$$

If F can be written as (2.6), it implies that the cost functions are convex.[6] Consequently, GANs parametrized with neural networks lead to non-monotone VIPs.

Assumption 3. *F is* monotone *and Ω is a compact convex set, such that* $\max_{\boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega} \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|^2 \leq R^2$.

[6]The convexity of the cost functions in (2.3) is a necessary condition (not sufficient) for the operator to be monotone. In the context of a zero-sum game, the convexity of the cost functions is a sufficient condition.

In that setting the quantity $g(\boldsymbol{\omega}) := \max_{\boldsymbol{\omega}\in\Omega} F(\boldsymbol{\omega}^*)^\top(\bar{\boldsymbol{\omega}}^* - \boldsymbol{\omega})$ is well defined and is equal to 0 if and only if $\boldsymbol{\omega}^*$ is a solution of (VIP). Moreover, if we are optimizing a *zero-sum game*, we have $\boldsymbol{\omega} = (\boldsymbol{\theta}, \boldsymbol{\varphi})$, $\Omega = \Theta \times \Phi$ and $F(\boldsymbol{\theta}, \boldsymbol{\varphi}) = [\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \quad - \nabla_{\boldsymbol{\varphi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi})]^\top$. Hence, the quantity $h(\boldsymbol{\theta}, \boldsymbol{\varphi}) := \max_{\boldsymbol{\theta}\in\Theta} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}^*) - \min_{\boldsymbol{\varphi}\in\Phi} \mathcal{L}(\boldsymbol{\theta}^*, \boldsymbol{\varphi})$ is well defined and equal to 0 if and only if $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$ is a *Nash equilibrium* of the game. The two functions g and h are called *merit functions* (more details on the concept of *merit functions* in §3). In the following, we call,

$$\mathrm{Err}(\boldsymbol{\omega}) := \begin{cases} \max\limits_{\boldsymbol{\theta}, \boldsymbol{\varphi}\in\Theta\times\Phi} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}^*) - \mathcal{L}(\boldsymbol{\theta}^*, \boldsymbol{\varphi}) & \text{if } F(\boldsymbol{\theta}, \boldsymbol{\varphi}) = [\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \ \ -\nabla_{\boldsymbol{\varphi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi})]^\top \\ \max\limits_{\boldsymbol{\omega}\in\Omega} F(\boldsymbol{\omega}^*)^\top(\bar{\boldsymbol{\omega}}^* - \boldsymbol{\omega}) & \text{otherwise.} \end{cases} \tag{4.2}$$

Averaging.. Alg. 1 (AvgSGD) presents the stochastic gradient method with *averaging*, which reduces to the standard (simultaneous) SGD updates for the two-player games used in the GAN literature, but returning an *average* of the iterates.

Theorem 2. *Under Assump. 1, 2 and 3, SGD with averaging (Alg. 1) with a constant step-size gives,*

$$\mathbb{E}[\mathrm{Err}(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{2\eta T} + \eta\frac{M^2 + \sigma^2}{2} \quad \textit{where} \quad \bar{\boldsymbol{\omega}}_T := \frac{1}{T}\sum_{t=0}^{T-1} \boldsymbol{\omega}_t\,, \quad \forall T \geq 1\,. \tag{4.3}$$

Thm. 2 uses a similar proof as [Nemirovski et al., 2009]. The constant term $\eta(M^2 + \sigma^2)/2$ in (4.3) is called the *variance term*. This type of bound is standard in stochastic optimization. We also provide in §6 a similar rate with an extra log factor when $\eta_t = \frac{\eta}{\sqrt{t}}$. We show that this variance term is smaller than the one of *SGD with prediction method* [Yadav et al., 2018] in §5.

Extrapolations.. Alg. 2 (AvgExtraSGD) adds an extrapolation step compared to Alg. 1 in order to reduce the oscillations due to the game between the two players. A theoretical consequence is that it has a smaller variance term than (4.3). As discussed previously, Assump. 2 made in Thm. 2 for the convergence of Alg. 1 is very strong in the unbounded setting. One advantage of SGD with *extrapolation* is that Thm. 3 does not require this assumption.

Theorem 3. *[Juditsky et al., 2011, Thm. 1] Under Assump. 1 and 3, if* $\mathbb{E}_\xi[F]$ *is L-Lipschitz, then SGD with* extrapolation and averaging *(Alg. 2) using a constant step-size* $\eta \leq \frac{1}{\sqrt{3}L}$ *gives,*

$$\mathbb{E}[\mathrm{Err}(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{\eta T} + \frac{7}{2}\eta\sigma^2 \quad \textit{where} \quad \bar{\boldsymbol{\omega}}_T := \frac{1}{T}\sum_{t=0}^{T-1} \boldsymbol{\omega}'_t\,, \quad \forall T \geq 1\,. \tag{4.4}$$

Since in practice $\sigma \ll M$, the variance term in (4.4) is significantly smaller than the one in (4.3). To summarize, SGD with *extrapolation* provides better convergence guarantees but requires two gradient computations and samples per iteration. This motivates our new method, Alg. 3 (AvgPastExtraSGD) which uses *extrapolation from the past* and achieves *the best of both worlds.*

Theorem 4. *Under Assump. 1 and 3, if $\mathbb{E}_\xi[F]$ is L-Lipschitz then SGD with* extrapolation from the past *using a constant step-size $\eta \leq \frac{1}{2\sqrt{3}L}$, gives that the averaged iterates converge as,*

$$\mathbb{E}[\mathrm{Err}(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{\eta T} + \frac{13}{2}\eta\sigma^2 \quad \textit{where} \quad \bar{\boldsymbol{\omega}}_T := \frac{1}{T}\sum_{t=0}^{T-1} \boldsymbol{\omega}'_t \quad \forall T \geq 1\,. \tag{4.5}$$

The bounds is similar to the one provided in Thm. 3 but each iteration of Alg. 3 is computationally half the cost of an iteration of Alg. 2.

5 Combining the techniques with established algorithms

In the previous sections we presented several techniques that converge on a simple bilinear example. These techniques can be combined in practice with existing algorithms. We propose to combine them to two standard algorithms used for training deep neural networks: the Adam optimizer [Kingma and Ba, 2015] and the SGD optimizer [Robbins and Monro, 1951]. Note that in the case of a two-player game (2.3), the previous results can be generalized to gradient updates with a different step-size for each player by simply rescaling the objectives $\mathcal{L}^{(\boldsymbol{\theta})}$ and $\mathcal{L}^{(\boldsymbol{\varphi})}$ by a different scaling factor. A detailed pseudo-code for Adam with extrapolation step (Extra-Adam) is given in Algorithm 4.

6 Related Work

The extragradient method is the standard algorithm to optimize variational inequalities. This algorithm has been originally introduced by Korpelevich [1976] and extended by Nesterov [2007] and Nemirovski [2004]. Stochastic versions of the extragradient have been recently analyzed [Juditsky et al., 2011, Yousefian et al., 2014, Iusem et al., 2017] for stochastic variational inequalities with *bounded*

Algorithm 4 Extra-Adam: proposed Adam with extrapolation step.

```
**input:** step-size $\eta$, decay rates for moment estimates $\beta_1, \beta_2$, access to the stochastic gradients $\nabla \ell_t(\cdot)$ and to the projection $P_\Omega[\cdot]$ onto the constraint set $\Omega$, initial parameter $\boldsymbol{\omega}_0$, averaging scheme $(\rho_t)_{t \geq 1}$
**for** $t = 0 \dots T-1$ **do**
    **Option 1: Standard extrapolation.**
        Sample new minibatch and compute stochastic gradient: $g_t \leftarrow \nabla \ell_t(\boldsymbol{\omega}_t)$
    **Option 2: Extrapolation from the past**
        Load previously saved stochastic gradient: $g_t = \nabla \ell_{t-1/2}(\boldsymbol{\omega}_{t-1/2})$
    Update estimate of first moment for extrapolation: $m_{t-1/2} \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
    Update estimate of second moment for extrapolation: $v_{t-1/2} \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
    Correct the bias for the moments: $\hat{m}_{t-1/2} \leftarrow m_{t-1/2}/(1-\beta_1^{2t-1})$, $\hat{v}_{t-1/2} \leftarrow v_{t-1/2}/(1-\beta_2^{2t-1})$
    Perform *extrapolation* step from iterate at time $t$: $\boldsymbol{\omega}_{t-1/2} \leftarrow P_\Omega[\boldsymbol{\omega}_t - \eta \frac{m_{t-1/2}}{\sqrt{v_{t-1/2}}+\epsilon}]$
    Sample new minibatch and compute stochastic gradient: $g_{t+1/2} \leftarrow \nabla \ell_{t+1/2}(\boldsymbol{\omega}_{t+1/2})$
    Update estimate of first moment: $m_t \leftarrow \beta_1 m_{t-1/2} + (1-\beta_1) g_{t+1/2}$
    Update estimate of second moment: $v_t \leftarrow \beta_2 v_{t-1/2} + (1-\beta_2) g_{t+1/2}^2$
    Compute bias corrected for first and second moment: $\hat{m}_t \leftarrow m_t/(1-\beta_1^{2t})$, $\hat{v}_t \leftarrow v_t/(1-\beta_2^{2t})$
    Perform *update* step from the iterate at time $t$: $\boldsymbol{\omega}_{t+1} \leftarrow P_\Omega[\boldsymbol{\omega}_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}]$
**end for**
**Output:** $\boldsymbol{\omega}_{T-1/2}$, $\boldsymbol{\omega}_T$ or $\bar{\boldsymbol{\omega}}_T = \sum_{t=0}^{T-1} \rho_{t+1} \boldsymbol{\omega}_{t+1/2} / \sum_{t=0}^{T-1} \rho_{t+1}$ (see (3.2) for online averaging)
```

constraints. A linearly convergent variance reduced version of the stochastic gradient method has been proposed by Palaniappan and Bach [2016] for strongly monotone variational inequalities.

Several methods to stabilize GANs consist in transforming a zero-sum formulation into a more general game that can no longer be cast as a saddle point problem. This is the case of the *non-saturating* formulation of GANs [Goodfellow et al., 2014, Fedus et al., 2018], the DCGANs [Radford et al., 2016], the *gradient penalty*[7] for WGANs [Gulrajani et al., 2017]. Yadav et al. [2018] propose an optimization method for GANs based on AltSGD using a momentum based step on the generator. Daskalakis et al. [2018] proposed a method inspired from game theory. Li et al. [2017] suggest to dualize the GAN objective to reformulate it as a maximization problem and Mescheder et al. [2017] propose to add the norm of the gradient in the objective and provide an interesting perspective on GANs, interpreting the

[7]The gradient penalty is only added to the discriminator cost function. Since this gradient penalty depends also on the generator, WGAN-GP cannot be cast as a SP problem and is actually a non-zero sum game.

training as the search of a two-player game equilibrium. A study of the continuous version of two player games has been conducted by Ratliff et al. [2016]. Interesting non-convex results were proved, for a new notion of regret minimization, by Hazan et al. [2017] and in the context of GANs by Grnarova et al. [2018].

The technique of *unrolling* steps proposed by Metz et al. [2017] can be confused with extrapolation but is actually fundamentally different: the perspective is to try to construct the "true generator objective function" unrolling for K steps the updates of the generator and then update the discriminator. Nevertheless the fact that this "true generator function" may not be found with a satisfying accuracy may lead to a different behavior than the one expected.

Regarding the averaging technique, some recent work appear to have already successfully used *geometric averaging* (3.1) for GANs in practice, but only briefly mention it [Karras et al., 2018, Mescheder et al., 2018]. By contrast the present work formally motivates and justifies the use of averaging for GANs by relating them to the VIP perspective, and sheds light on its underlying intuitions in §3.1. Another independent work [Yazıcı et al., 2019] made a similar attempt but in the context of regret minimization in games. Mertikopoulos et al. [2019] also independently explored extrapolation providing asymptotic convergence results (i.e. without any rate of convergence) in the context of *coherent saddle point*. The coherence assumption is slightly weaker than monotonicity.

7 Experiments

Our goal in this experimental section is not to provide new state-of-the art results with architectural improvements or a new GAN formulation but to show that using the *techniques* (with theoretical guarantees in the monotone case) that we introduced earlier allow us to optimize standard GANs in a better way. These techniques, which are orthogonal to the design of new formulations of GAN optimization objectives, and to architectural choices, can potentially be used for the training of any type of GAN. We will compare the following optimization algorithms: baselines are SGD and Adam using either simultaneous updates on the generator and on the discriminator (denoted **SimAdam** and **SimSGD**) or k updates on the discriminator alternated with 1 update on the generator (denoted **AltSGD**$\{k\}$ and **AltAdam**$\{k\}$)[8]. Variants that use *extrapolation* are denoted **ExtraSGD** (Alg. 2) and **ExtraAdam** (Alg. 4). Variants using *extrapolation from the past* are **PastExtraSGD** (Alg. 3) and **PastExtraAdam** (Alg. 4). We also present results

[8]In the original WGAN [Arjovsky et al., 2017] paper the authors use $k = 5$.

using as output the *averaged* iterates, adding **Avg** as a prefix of the algorithm name when we use (uniform) *averaging.*

7.1 Bilinear saddle point (stochastic)

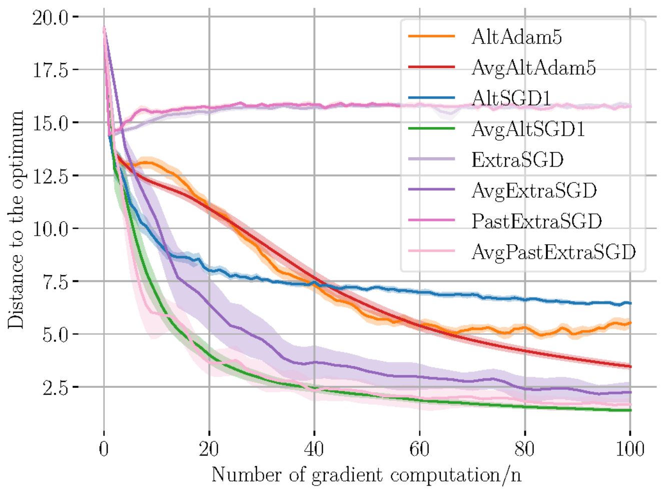


Figure 4.3: Performance of the considered stochastic optimization algorithms on the bilinear problem (7.1). Each method uses its respective optimal step-size found by grid-search.

We evaluate the performance of the various stochastic algorithms first on a simple ($n = 10^3, d = 10^3$) finite sum bilinear objective (a monotone operator) constrained to $[-1, 1]^d$:

$$\frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{\theta}^\top \boldsymbol{M}^{(i)}\boldsymbol{\varphi} + \boldsymbol{\theta}^\top \boldsymbol{a}^{(i)} + \boldsymbol{\varphi}^\top \boldsymbol{b}^{(i)}\right) \tag{7.1}$$

$$\text{solved by } (\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*) \text{ s.t.} \begin{cases} \bar{\boldsymbol{M}}\boldsymbol{\varphi}^* = -\bar{\boldsymbol{a}} \\ \bar{\boldsymbol{M}}^T\boldsymbol{\theta}^* = -\bar{\boldsymbol{b}} \end{cases},$$

where $\bar{\boldsymbol{a}} := \frac{1}{n}\sum_{i=1}^n \boldsymbol{a}^{(i)}$, $\bar{\boldsymbol{b}} := \frac{1}{n}\sum_{i=1}^n \boldsymbol{b}^{(i)}$ and $\bar{\boldsymbol{M}} := \frac{1}{n}\sum_{i=1}^n \boldsymbol{M}^{(i)}$. Marices and vectors $\boldsymbol{M}_{kj}^{(i)}$, $\boldsymbol{a}_k^{(i)}$, $\boldsymbol{b}_k^{(i)}$; $1 \leq i \leq n$, $1 \leq j, k \leq d$ were randomly generated, but ensuring that $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$ would belong to $[-1, 1]^d$. Results are shown in Fig. 4.3. We can see that AvgSGD and AvgPastExtraSGD perform the best on this task.

7.2 WGAN and WGAN-GP on CIFAR10

We now evaluate the proposed techniques in the context of GAN training, which is a challenging stochastic optimization problem where the objectives of both players are

Model	WGAN (DCGAN)			WGAN-GP (ResNet)	
Method	no averaging	uniform avg	EMA	no averaging	uniform avg
SimAdam	*6.05 ± .12*	5.83 ± .16	6.08 ± .10	7.54 ± .21	7.74 ± .27
AltAdam5	*5.45 ± .08*	5.72 ± .06	5.49 ± .05	7.20 ± .06	7.67 ± .15
ExtraAdam	**6.38 ± .09**	**6.38 ± .20**	**6.37 ± .08**	7.79 ± .09	**8.26 ± .12**
PastExtraAdam	5.98 ± .15	6.07 ± .19	6.01 ± .11	7.71 ± .12	7.84 ± .18
OptimAdam	5.74 ± .10	5.80 ± .08	5.78 ± .05	7.80 ± .07	7.99 ± .12

Table 4.1: Best inception scores (averaged over 5 runs) achieved on CIFAR10 for every considered Adam variant. OptimAdam is the related *Optimistic Adam* [Daskalakis et al., 2018] algorithm. EMA denotes *exponential moving average* (with $\beta = 0.999$, see Eq. 3.2). We see that the techniques of extrapolation and averaging consistently enable improvements over the baselines (in italic).

non-convex. We propose to evaluate the Adam variants of the different optimization algorithms (see Alg. 4 for Adam with *extrapolation*) by training two different architectures on the CIFAR10 dataset [Krizhevsky and Hinton, 2009]. First we consider a constrained zero-sum game by training the DCGAN architecture [Radford et al., 2016] with the WGAN objective and weight clipping as proposed in Arjovsky et al. [2017]. Then we compared the different methods on a state of the art architecture, by training a ResNet with the WGAN-GP objective similar to Gulrajani et al. [2017]. Models are evaluated using the inception score [Salimans et al., 2016] computed on 10,000 samples.

For each algorithm we did an extensive search over the hyperparameters of Adam, we fixed $\beta_1 = 0.5$ and $\beta_2 = 0.9$ for all methods as they seemed to perform well. We want to emphasize that as proposed by Heusel et al. [2017] it is quite important to set different learning rate for the generator and the discriminator. Each experiments were run with 5 different random seeds for 500,000 updates of the generator.

Table 4.1 reports the best inception score achieved on this problem by each considered method. We see that the techniques of *extrapolation* and *averaging* consistently enable improvements over the baselines (see §7.4 for more experiments on *averaging*). Fig. 4.4 shows training curves for each method (for their best performing learning rate), as well as samples from an ExtraAdam-trained WGAN. For both tasks, using an *extrapolation step* and averaging with Adam (ExtraAdam) outperformed all other methods. ExtraAdam with averaging is able to reach state of the art results on CIFAR10 similar to Miyato et al. [2018]. We also observed that methods based on *extrapolation* are less sensitive to the choice of learning rate and can be used with higher learning rates with less degradation; see App. §7.3 for more details.

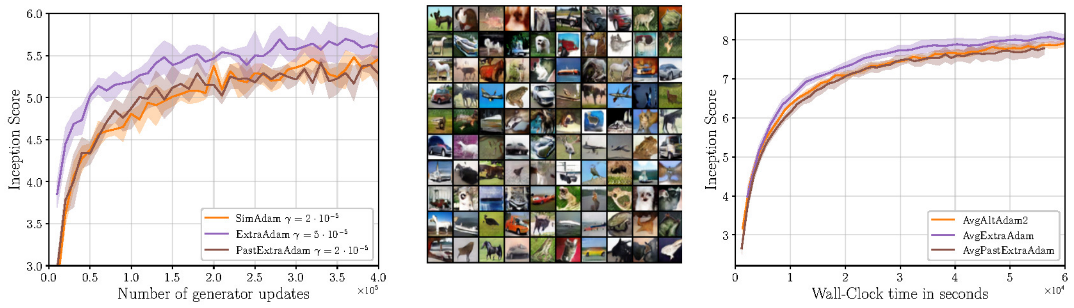


Figure 4.4: Left: Mean and standard deviation of the inception score computed over 5 runs for each method on WGAN trained on CIFAR10. To keep the graph readable we show only SimAdam but AltAdam performs similarly. **Middle:** Samples from a ResNet generator trained with the WGAN-GP objective using AvgExtraAdam. **Right:** WGAN-GP trained on CIFAR10: mean and standard deviation of the inception score computed over 5 runs for each method using the best performing learning rate plotted over wall-clock time; all experiments were run on a NVIDIA Quadro GP100 GPU. We see that ExtraAdam converges faster than the Adam baselines.

8 Conclusion

We newly addressed GAN objectives in the framework of variational inequality. We tapped into the optimization literature to provide more principled sound techniques to optimize such games. We leveraged these techniques to develop practical optimization algorithms suitable for a wide range of GAN training objectives (including non-zero sum games and projections onto constraints). We experimentally verified that this could yield better trained models, achieving to our knowledge the best inception score when optimizing a WGAN objective on the reference unmodified DCGAN architecture [Radford et al., 2016]. The presented techniques address a fundamental problem in GAN training in a principled way, and are orthogonal to the design of new GAN architectures and objectives. They are thus likely to be widely applicable, and benefit future development of GANs.

5 Prologue to the Second Contribution

1 Article Details

A Closer Look at the Optimization Landscapes of Generative Adversarial Networks. *Hugo Berard*, Gauthier Gidel*, Amjad Almahairi, Pascal Vincent and Simon Lacoste-Julien.* This paper was published at ICLR 2020 [Berard et al., 2020].

*Equal contribution.

2 Contributions of the authors

The idea for this paper emerged from frequent meeting and discussion between Hugo Berard and Gauthier Gidel, where they discussed and tried to understand the challenges and instabilities of training GANs. Hugo Berard contributed his knowledge and experience with GANs and deep learning, while Gauthier Gidel contributed his knowledge and expertise of optimization. They both contributed to the writting of the paper with Hugo Berard focusing more on the visualization tools and the numerical experiments, and Gauthier Gidel focusing more on the theoretical results and the theoretical justifications for the visualizations and observations. Amjad Almahairi also contributed to the writing of the paper and to the experiments. Pascal Vincent and Simon Lacoste-Julien supervised the project.

3 Context and Impact

At the time of this article, while there was some theoretical results about smooth games optimization, a lot of questions were still open and the assumptions were often unrealistic. In particular, there was almost no results about non-monotone variational inequality and games. Inspired by empirical work on understanding the loss landscape of deep neural networks [Goodfellow et al., 2015b], we tried to adopt a similar empirical approach to study GANs. We decided to focus our attention on trying to evaluate the properties of the stationary points of GANs. The main observation we made was to observe that actually GANs often do not converge to Nash-Equilibria contrary to popular beliefs. This lead several works to consider new notions of equilibria for smooth games Fiez et al. [2020], Jin et al. [2020].

Limitations and Remarks. There are several limitations to this work. First in our work we consider that a stationary point has been reached when the gradient norm is sufficiently small, however in some cases this gradient norm is still relatively high which raises the question whether we have indeed converge to a stationary point. This observation was also made in Jelassi et al. [2021] where they observe that models that achieve good results often do not converge to very low gradient norm solutions. Furthermore we only consider the stationary points extragradient converge to, this might not be representative of the full landscape of GANs and might be biased to a certain type of stationary points. We tried to use SGDA, but SGDA was too unstable and was often not converging. Finally most of the observation we make are local, they only look at small neighborhood around the solution, and do not tell us about the rest of the landscape of GANs outside of the points the GAN has converged too, in particular while we observe that extragradient in GANs does not converge to Nash-Equilibria, Nash-Equilibria might still exist.

6 A Closer Look at the Optimization Landscapes of Generative Adversarial Networks

Abstract

Generative adversarial networks have been very successful in generative modeling, however they remain relatively challenging to train compared to standard deep neural networks. In this paper, we propose new visualization techniques for the optimization landscapes of GANs that enable us to study the game vector field resulting from the concatenation of the gradient of both players. Using these visualization techniques we try to bridge the gap between theory and practice by showing empirically that the training of GANs exhibits significant rotations around Local Stable Stationary Points (LSSP), similar to the one predicted by theory on toy examples. Moreover, we provide empirical evidence that GAN training converges to a *stable* stationary point which is a saddle point for the generator loss, not a minimum, while still achieving excellent performance.[1]

1 Introduction

Deep neural networks have exhibited remarkable success in many applications [Krizhevsky et al., 2012]. This success has motivated many studies of their non-convex loss landscape [Choromanska et al., 2015, Kawaguchi, 2016, Li et al., 2018b], which, in turn, has led to many improvements, such as better initialization and optimization methods [Glorot and Bengio, 2010, Kingma and Ba, 2015].

While most of the work on studying non-convex loss landscapes has focused on single objective minimization, some recent class of models require the joint minimization of several objectives, making their optimization landscape intrinsically different. Among these models is the generative adversarial network (GAN) [Goodfellow et al., 2014] which is based on a two-player game formulation and has achieved

[1]Code available at `https://bit.ly/2kwTu87`

state-of-the-art performance on some generative modeling tasks such as image generation [Brock et al., 2019].

On the theoretical side, many papers studying multi-player games have argued that one main optimization issue that arises in this case is the rotation due to the adversarial component of the game [Mescheder et al., 2018, Balduzzi et al., 2018, Gidel et al., 2019b]. This has been extensively studied on toy examples, in particular on the so-called bilinear example [Goodfellow, 2016] (a.k.a Dirac GAN [Mescheder et al., 2018]). However, those toy examples are very far from the standard realistic setting of image generation involving deep networks and challenging datasets. To our knowledge it remains an open question if this rotation phenomenon actually occurs when training GANs in more practical settings.

In this paper, we aim at closing this gap between theory and practice. Following Mescheder et al. [2017] and Balduzzi et al. [2018], we argue that instead of studying the loss surface, we should study the *game vector field* (i.e., the concatenation of each player's gradient), which can provide better insights to the problem. To this end, we propose a new visualization technique that we call *Path-angle* which helps us observe the nature of the game vector field close to a stationary point for high dimensional models, and carry on an empirical investigation of the properties of the optimization landscape of GANs. The core questions we want to address may be summarized as the following:

Is rotation a phenomenon that occurs when training GANs on real world datasets, and do existing training methods find local Nash equilibria?

To answer this question we conducted extensive experiments by training different GAN formulations (NSGAN and WGAN-GP) with different optimizers (Adam and ExtraAdam) on three datasets (MoG, MNIST and CIFAR10). Based on our experiments and using our visualization techniques we observe that the landscape of GANs is fundamentally different from the standard loss surfaces of deep networks. Furthermore, we provide evidence that existing GAN training methods do not converge to a local Nash equilibrium.

Contributions. More precisely, our contributions are the following: (i) We propose studying empirically the game vector field (as opposed to studying the loss surfaces of each player) to understand training dynamics in GANs using a novel visualization tool, which we call *Path-angle* and that captures the rotational and attractive behaviors near local stationary points (ref. §4.2). (ii) We observe experimentally on both a mixture of Gaussians, MNIST and CIFAR10 datasets that a variety of GAN formulations have a significant rotational behavior around their locally stable stationary points (ref. §5.1). (iii) We provide empirical evidence that existing training procedures find stable stationary points that are saddle points, not minima, for the loss function of the generator (ref. § 5.2).

2 Related work

Improving the training of GANs has been an active research area in the past few years. Most efforts in stabilizing GAN training have focused on formulating new objectives [Arjovsky et al., 2017], or adding regularization terms [Gulrajani et al., 2017, Mescheder et al., 2017, 2018]. In this work, we try to characterize the difference in the landscapes induced by different GAN formulations and how it relates to improving the training of GANs.

Recently, Nagarajan and Kolter [2017], Mescheder et al. [2018] show that a local analysis of the eigenvalues of the Jacobian of the game can provide guarantees on local stability properties. However, their theoretical analysis is based on some unrealistic assumptions such as the generator's ability to fully capture the real distribution. In this work, we assess experimentally to what extent these theoretical stability results apply in practice.

Rotations in differentiable games have been mentioned and interpreted by [Mescheder et al., 2018, Balduzzi et al., 2018] and Gidel et al. [2019b]. While these papers address rotations in games from a theoretical perspective, it was never shown that GANs, which are games with highly non-convex losses, suffered from these rotations in practice. To our knowledge, trying to quantify that GANs actually suffer from this rotational component in practice for real world dataset is novel.

The stable points of the gradient dynamics in general games have been studied independently by Mazumdar et al. [2020] and Adolphs et al. [2018]. They notice that the locally stable stationary point of some games are not local Nash equilibria. In order to reach a local Nash equilibrium, Adolphs et al. [2018], Mazumdar et al. [2019] develop techniques based on second order information. In this work, we argue that reaching local Nash equilibria may not be as important as one may expect and that we do achieve good performance at a locally stable stationary point.

Several works have studied the loss landscape of deep neural networks. Goodfellow et al. [2015b] proposed to look at the linear path between two points in parameter space and show that neural networks behave similarly to a convex loss function along this path. Draxler et al. [2018] proposed an extension where they look at nonlinear paths between two points and show that local minima are connected in deep neural networks. Another extension was proposed by [Li et al., 2018a] where they use contour plots to look at the 2D loss surface defined by two directions chosen appropriately. In this paper, we use a similar approach of following the linear path between two points to gain insight about GAN optimization landscapes. However, in this context, looking at the loss of both players along that path may be uninformative. We propose instead to look, along a linear path from initialization to best solution, at the game vector field, particularly at its angle w.r.t. the linear path, the *Path-angle*.

Another way to gain insight into the landscape of deep neural networks is by looking at the Hessian of the loss; this was done in the context of single objective minimization by [Dauphin et al., 2014, Sagun et al., 2016, 2017, Alain et al., 2019]. Compared to linear path visualizations which can give global information (but only along one direction), the Hessian provides information about the loss landscape in several directions but only locally. The full Hessian is expensive to compute and one often has to resort to approximations such has computing only the top-k eigenvalues. While, the Hessian is symmetric and thus has real eigenvalues, the Jacobian of a game vector field is significantly different since it is in general not symmetric, which means that the eigenvalues belong to the complex plane. In the context of GANs, Mescheder et al. [2017] introduced a gradient penalty and use the eigenvalues of the Jacobian of the game vector field to show its benefits in terms of stability. In our work, we compute these eigenvalues to assess that, on different GAN formulations and datasets, existing training procedures find a locally stable stationary point that is a saddle point for the loss function of the generator.

3 Formulations for GAN optimization and their practical implications

3.1 The standard game theory formulation

From a game theory point of view, GAN training may be seen as a game between two players: the discriminator $D_{\boldsymbol{\varphi}}$ and the generator $G_{\boldsymbol{\theta}}$, each of which is trying to minimize its loss $\mathcal{L}_D$ and $\mathcal{L}_G$, respectively. Using the same formulation as Mescheder et al. [2017], the GAN objective takes the following form (for simplicity of presentation, we focus on the unconstrained formulation):

$$\boldsymbol{\theta}^* \in \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \mathbb{R}^p} \mathcal{L}_G(\boldsymbol{\theta}, \boldsymbol{\varphi}^*) \qquad \text{and} \qquad \boldsymbol{\varphi}^* \in \operatorname*{arg\,min}_{\boldsymbol{\varphi} \in \mathbb{R}^d} \mathcal{L}_D(\boldsymbol{\theta}^*, \boldsymbol{\varphi}) \,. \tag{3.1}$$

The solution $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$ is called a *Nash equilibrium* (NE). In practice, the considered objectives are non-convex and we typically cannot expect better than a *local* Nash equilibrium (LNE), i.e. a point at which (3.1) is only locally true (see e.g. [Adolphs et al., 2018] for a formal definition). Ratliff et al. [2016] derived some derivative-based necessary and sufficient conditions for being a LNE. They show that, for being a local NE it is sufficient to be a *differential Nash equilibrium*:

Definition 1 (Differential NE)**.** *A point* $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$ *is a* differential Nash equilibrium *(DNE) iff*

$$\|\nabla_{\boldsymbol{\theta}} \mathcal{L}_G(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)\| = \|\nabla_{\boldsymbol{\varphi}} \mathcal{L}_D(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)\| = 0\,,\ \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_G(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*) \succ 0 \text{ and } \nabla^2_{\boldsymbol{\varphi}} \mathcal{L}_D(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*) \succ 0 \tag{3.2}$$

where $\boldsymbol{S} \succ 0$ *if and only if* $\boldsymbol{S}$ *is positive definite.*

Being a DNE is not necessary for being a LNE because a local Nash equilibrium may have Hessians that are only semi-definite. NE are commonly used in GANs to describe the goal of the learning procedure [Goodfellow et al., 2014]: in this definition, $\boldsymbol{\theta}^*$ (resp. $\boldsymbol{\varphi}^*$) is seen as a local minimizer of $\mathcal{L}_G(\cdot, \boldsymbol{\varphi}^*)$ (resp. $\mathcal{L}_D(\boldsymbol{\theta}^*, \cdot)$).

Under this view, however, the interaction between the two networks is not taken into account. This is an important aspect of the game stability that is missed in the definition of DNE (and Nash equilibrum in general). We illustrate this point in the following section, where we develop an example of a game for which gradient methods converge to a point which is a saddle point for the generator's loss and thus not a DNE for the game.

3.2 An alternative formulation based on the game vector field

In practice, GANs are trained using first order methods that compute the gradients of the losses of each player. Following Gidel et al. [2019a], an alternative point of view on optimizing GANs is to jointly consider the players' parameters $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ as a joint state $\boldsymbol{\omega} := (\boldsymbol{\theta}, \boldsymbol{\varphi})$, and to study the vector field associated with these gradients,[2] which we call the *game vector field*

$$\boldsymbol{v}(\boldsymbol{\omega}) := \begin{bmatrix} \nabla_{\boldsymbol{\theta}} \mathcal{L}_G(\boldsymbol{\omega})^\top & \nabla_{\boldsymbol{\varphi}} \mathcal{L}_D(\boldsymbol{\omega})^\top \end{bmatrix}^\top \quad \text{where} \quad \boldsymbol{\omega} := (\boldsymbol{\theta}, \boldsymbol{\varphi})\,. \tag{3.3}$$

With this perspective, the notion of DNE is replaced by the notion of locally stable stationary point (LSSP). Verhulst [1989, Theorem 7.1] defines a LSSP $\boldsymbol{\omega}^*$ using the eigenvalues of the Jacobian of the game vector field $\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$ at that point.

[2]Note that, in practice, the joint vector field (3.3) is *not* a gradient vector field, i.e., it cannot be rewritten as the gradient of a single function.

Zero-sum game	Non-zero-sum game
NE $\Rightarrow$ LSSP [Mescheder et al., 2018]	NE $\not\Rightarrow$ LSSP (Example 2, §1.2)
NE $\not\Leftarrow$ LSSP [Adolphs et al., 2018]	NE $\not\Leftarrow$ LSSP (Example 1)

Table 6.1: Summary of the implications between Differentiable Nash Equilibrium (DNE) and a locally stable stationnary point (LSSP): in general, being a DNE is neither necessary or sufficient for being a LSSP.

Definition 2 (LSSP)**.** *A point $\boldsymbol{\omega}^*$ is a* locally stable stationary point *(LSSP) iff*

$$\boldsymbol{v}(\boldsymbol{\omega}^*) = 0 \qquad \textit{and} \qquad \Re(\lambda) > 0\,, \quad \forall \lambda \in \mathrm{Sp}(\nabla \boldsymbol{v}(\boldsymbol{\omega}^*))\,. \tag{3.4}$$

where $\Re$ denote the real part of the eigenvalue λ belonging to the spectrum of $\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$.

This definition is not easy to interpret but one can intuitively understand a LSSP as a stationary point (a point $\boldsymbol{\omega}^*$ where $\boldsymbol{v}(\boldsymbol{\omega}^*) = 0$) to which all neighbouring points are attracted. We will formalize this intuition of attraction in Proposition 1. In our two-player game setting, the Jacobian of the game vector field around the LSSP has the following block-matrices form:

$$\nabla \boldsymbol{v}(\boldsymbol{\omega}^*) = \begin{bmatrix} \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_G(\boldsymbol{\omega}^*) & \nabla_{\boldsymbol{\varphi}} \nabla_{\boldsymbol{\theta}} \mathcal{L}_G(\boldsymbol{\omega}^*) \\ \nabla_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\varphi}} \mathcal{L}_D(\boldsymbol{\omega}^*) & \nabla^2_{\boldsymbol{\varphi}} \mathcal{L}_D(\boldsymbol{\omega}^*) \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_1 & \boldsymbol{B} \\ \boldsymbol{A} & \boldsymbol{S}_2 \end{bmatrix}. \tag{3.5}$$

When $\boldsymbol{B} = -\boldsymbol{A}^\top$, being a DNE is a sufficient condition for being of LSSP [Mazumdar et al., 2020]. However, some LSSP may not be DNE [Adolphs et al., 2018], meaning that the optimal generator $\boldsymbol{\theta}^*$ could be a saddle point of $\mathcal{L}_G(\cdot, \boldsymbol{\varphi}^*)$, while the optimal joint state $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$ may be a LSSP of the game. We summarize these properties in Table 6.1. In order to illustrate the intuition behind this counter-intuitive fact, we study a simple example where the generator is 2D and the discriminator is 1D.

Example 1. *Let us consider $\mathcal{L}_G$ as a hyperbolic paraboloid (a.k.a., saddle point function) centered in $(1,1)$ where $(1,\varphi)$ is the principal descent direction and $(-\varphi, 1)$ is the principal ascent direction, while $\mathcal{L}_D$ is a simple bilinear objective.*

$$\mathcal{L}_G(\theta_1, \theta_2, \varphi) = (\theta_2 - \varphi\theta_1 - 1)^2 - \tfrac{1}{2}(\theta_1 + \varphi\theta_2 - 1)^2\,, \;\; \mathcal{L}_D(\theta_1, \theta_2, \varphi) = \varphi(5\theta_1 + 4\theta_2 - 9)$$

We plot $\mathcal{L}_G$ in Fig. 6.1b. Note that the discriminator φ controls the principal descent direction of $\mathcal{L}_G$.

We show (see § 1.2) that $(\theta_1^*, \theta_2^*, \varphi^*) = (1, 1, 0)$ is a locally stable stationary point but is not a DNE: the generator loss at the optimum $(\theta_1, \theta_2) \mapsto \mathcal{L}_G(\theta_1, \theta_2, \varphi^*) = \theta_2^2 - \frac{1}{2}\theta_1^2$ is not at a DNE because it has a clear descent direction, $(1, 0)$. However, if the generator follows this descent direction, the dynamics will remain stable because the discriminator will update its parameter, rotating the saddle and making $(1, 0)$

an ascent direction. We call this phenomenon *dynamic stability*: the loss $\mathcal{L}_G(\cdot, \varphi^*)$ is unstable for a fixed φ^* but becomes stable when φ dynamically interacts with the generator around φ^*.

A mechanical analogy for this dynamic stability phenomenon is a ball in a rotating saddle—even though the gravity pushes the ball to escape the saddle, a quick enough rotation of the saddle would trap the ball at the center (see [Thompson et al., 2002] for more details). This analogy has been used to explain Paul's trap [Paul, 1990]: a counter-intuitive way to trap ions using a dynamic electric field. In Example 1, the parameter φ explicitly controls the rotation of the saddle.

This example illustrates the fact that the DNE corresponds to a notion of *static stability*: it is the stability of one player's loss given the other player is fixed. Conversely, LSSP captures a notion of *dynamic stability* that considers both players jointly.

By looking at the game vector field we capture these interactions. Fig. 6.1b only captures a snapshot of the generator's loss surface for a fixed φ and indicates static instability (the generator is at a saddle point of its loss). In Fig. 6.1a, however, one can see that, starting from any point, we will rotate around the stationary point $(\varphi^*, \theta_1^*) = (0, 1)$ and eventually converge to it.

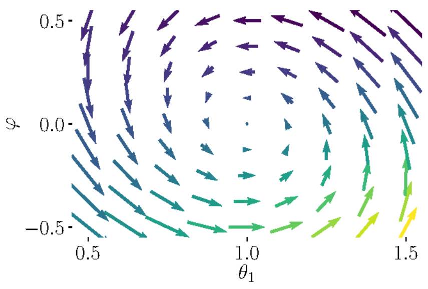
The visualization of the game vector field reveals an interesting behavior that does not occur in single objective minimization: close to a LSSP, the parameters rotate around it. Understanding this phenomenon is key to grasp the optimization difficulties arising in games. In the next section, we formally characterize the notion of rotation around a LSSP and in §4 we develop tools to visualize it in high dimensions. Note that gradient methods may converge to saddle points in single objective minimization, but these are not *stable* stationary points, unlike in our game example.

3.3 Rotation and attraction around locally stable stationary points in games

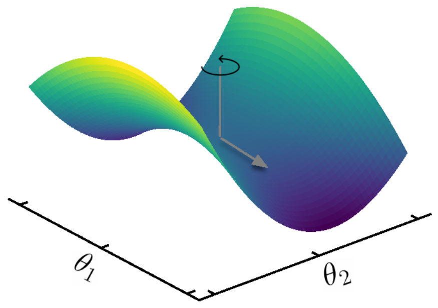
In this section, we formalize the notions of rotation and attraction around LSSP in games, which we believe may explain some difficulties in GAN training. The local stability of a LSSP is characterized by the eigenvalues of the Jacobian $\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$ because we can linearize $\boldsymbol{v}(\boldsymbol{\omega})$ around $\boldsymbol{\omega}^*$:

$$\boldsymbol{v}(\boldsymbol{\omega}) \approx \nabla \boldsymbol{v}(\boldsymbol{\omega}^*)(\boldsymbol{\omega} - \boldsymbol{\omega}^*). \tag{3.6}$$

If we assume that (3.6) is an equality, we have the following theorem.



(a) 2D projection of the vector field.



(b) Landscape of the generator loss.

Figure 6.1: Visualizations of Example 1. Left: projection of the game vector field on the plane $\theta_2 = 1$. Right: Generator loss. The descent direction is $(1, \varphi)$ (in grey). As the generator follows this descent direction, the discriminator changes the value of φ, making the saddle rotate, as indicated by the circular black arrow.

Proposition 1. *Let us assume that (3.6) is an equality and that $\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$ is diagonalizable, then there exists a basis $\boldsymbol{P}$ such that the coordinates $\tilde{\boldsymbol{\omega}}_j(t) := [\boldsymbol{P}(\boldsymbol{\omega}(t) - \boldsymbol{\omega}^*)]_j$ where $\boldsymbol{\omega}(t)$ is a solution of (3.6) have the following behavior: for $\lambda_j \in \operatorname{Sp} \nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$ we have,*

1. *If $\lambda_j \in \mathbb{R}$, we observe* pure attraction*:* $\quad \tilde{\boldsymbol{\omega}}_j(t) = e^{-\lambda_j t}\tilde{\boldsymbol{\omega}}_j(0)\,.$

2. *If $\Re(\lambda_j) = 0$, we observe* pure rotation*:*

$$\begin{bmatrix} \tilde{\boldsymbol{\omega}}_j(t) \\ \tilde{\boldsymbol{\omega}}_{j+1}(t) \end{bmatrix} = \begin{bmatrix} \cos|\lambda_j t| & \sin|\lambda_j t| \\ -\sin|\lambda_j t| & \cos|\lambda_j t| \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\omega}}_j(0) \\ \tilde{\boldsymbol{\omega}}_{j+1}(0) \end{bmatrix}$$

3. *Otherwise, we observe both:*

$$\begin{bmatrix} \tilde{\boldsymbol{\omega}}_j(t) \\ \tilde{\boldsymbol{\omega}}_{j+1}(t) \end{bmatrix} = e^{-\operatorname{Re}(\lambda_j)t} \begin{bmatrix} \cos \operatorname{Im}(\lambda_j t) & \sin \operatorname{Im}(\lambda_j t) \\ -\sin \operatorname{Im}(\lambda_j t) & \cos \operatorname{Im}(\lambda_j t) \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\omega}}_j(0) \\ \tilde{\boldsymbol{\omega}}_{j+1}(0) \end{bmatrix}$$

Note that we re-ordered the eigenvalues such that the complex conjugate eigenvalues form pairs: if $\lambda_j \notin \mathbb{R}$ then $\lambda_{j+1} = \bar{\lambda}_j$.

Matrices in 2. and 3. are rotations matrices. They induce a rotational behavior illustrated in Fig 6.1a.

This proposition shows that the dynamics of $\boldsymbol{\omega}(t)$ can be decomposed in a particular basis into attractions and rotations over components that do not interact between each other. Rotation does not appear in single objective minimization around a local minimum, because the eigenvalues of the Hessian of the objective are always real. Mescheder et al. [2017] discussed that difficulties in training GANs may be a result of the imaginary part of the eigenvalues of the Jacobian of the game vector field and Gidel et al. [2019b] mentioned that games have a natural oscillatory behavior. This cyclic behavior has been explained in [Balduzzi et al., 2018] by a non-zero Hamiltonian component in the Helmholtz decomposition of the Jacobian of the game vector field. All these explanations are related to the spectral properties of this Jacobian. The goal of Proposition 1 is to provide a formal definition to the notions of *rotation* and *attraction* we are dealing with in this paper.

In the following section, we introduce a new tool in order to assess the magnitude of the rotation around a LSSP compared to the attraction to this point.

4 Visualization for the vector field landscape

Neural networks are parametrized by a large number of variables and visualizations are only possible using low dimensional plots (1D or 2D). We first present a standard visualization tool for deep neural network loss surfaces that we will exploit in §4.2.

4.1 Standard visualizations for the loss surface

One way to visualize a neural network's loss landscape is to follow a parametrized path $\boldsymbol{\omega}(\alpha)$ that connects two parameters $\boldsymbol{\omega}, \boldsymbol{\omega}'$ (often one is chosen early in learning and another one is chosen late in learning, close to a solution). A path is a continuous function $\boldsymbol{\omega}(\cdot)$ such that $\boldsymbol{\omega}(0) = \boldsymbol{\omega}$ and $\boldsymbol{\omega}(1) = \boldsymbol{\omega}'$. Goodfellow et al. [2015b] considered a linear path $\boldsymbol{\omega}(\alpha) = \alpha\boldsymbol{\omega} + (1-\alpha)\boldsymbol{\omega}'$. More complex paths can be considered to assess whether different minima are connected [Draxler et al., 2018].

4.2 Proposed visualization: Path-angle

We propose to study the linear path between parameters early in learning and parameters late in learning. We illustrate the extreme cases for the game vector field along this path in simple examples in Figure 6.2(a-c): pure attraction occurs when the vector field perfectly points to the optimum (Fig. 6.2a) and pure rotation when the vector field is orthogonal to the direction to the optimum (Fig. 6.2b).

In practice, we expect the vector field to be in between these two extreme cases (Fig. 6.2c). In order to determine in which case we are, around a LSSP, in practice, we propose the following tools.

Path-norm.. We first ensure that we are in a neighborhood of a stationary point by computing the norm of the vector field. Note that considering independently the norm of each player may be misleading: even though the gradient of one player may be close to zero, it does not mean that we are at a stationary point since the other player might still be updating its parameters. **Path-angle..** Once we are close to a final point $\boldsymbol{\omega}'$, i.e., in a neighborhood of a LSSP, we propose to look at the angle between the vector field (3.3) and the linear path from $\boldsymbol{\omega}$ to $\boldsymbol{\omega}'$. Specifically, we monitor the cosine of this angle, a quantity we call *Path-angle*:

$$c(\alpha) := \frac{\langle \boldsymbol{\omega}' - \boldsymbol{\omega}, \boldsymbol{v}_\alpha \rangle}{\|\boldsymbol{\omega}' - \boldsymbol{\omega}\| \|\boldsymbol{v}_\alpha\|} \quad \text{where} \quad \boldsymbol{v}_\alpha := \boldsymbol{v}(\alpha\boldsymbol{\omega}' + (1-\alpha)\boldsymbol{\omega}) \, , \; \alpha \in [a, b] \, . \tag{4.1}$$

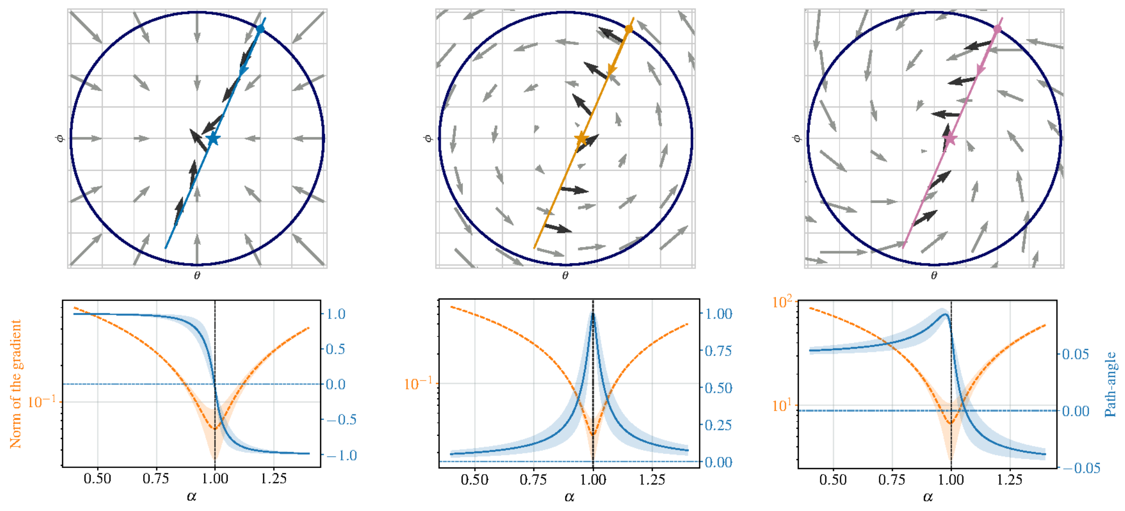
Usually $[a, b] = [0, 1]$, but since we are interested in the landscape around a LSSP, it might be more informative to also consider further extrapolated points around $\boldsymbol{\omega}'$ with $b > 1$.

Eigenvalues of the Jacobian.. Another important tool to gain insights on the behavior close to a LSSP, as discussed in §3.2, is to look at the eigenvalues of $\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$. We propose to compute the top-k eigenvalues of this Jacobian. When all the eigenvalues have positive real parts, we conclude that we have reached a LSSP, and if some eigenvalues have large imaginary parts, then the game has a strong rotational behavior (Thm. 1). Similarly, we can also compute the top-k eigenvalues of the diagonal blocks of the Jacobian, which correspond to the Hessian of each player. These eigenvalues can inform us on whether we have converged to a LSSP that is not a LNE.

An important advantage of the Path-angle relative to the computation of the eigenvalues of $\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$ is that it only requires computing gradients (and not second order derivatives, which may be prohibitively computationally expensive for deep networks). Also, it provides information along a whole path between two points and thus, more global information than the Jacobian computed at a single point. In the following section, we use the Path-angle to study the archetypal behaviors presented in Thm 1.

4.3 Archetypal behaviors of the Path-angle around a LSSP

Around a LSSP, we have seen in (3.6) that the behavior of the vector field is mainly dictated by the Jacobian matrix $\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$. This motivates the study of the behavior



(a) Attraction only **(b)** Rotation only **(c)** Rotation and attraction

Figure 6.2: Above: game vector field (in grey) for different archetypal behaviors. The equilibrium of the game is at $(0, 0)$. Black arrows correspond to the directions of the vector field at different linear interpolations between two points: $\bullet$ and $\star$. **Below**: path-angle $c(\alpha)$ for different archetypal behaviors (right y-axis, in blue). The left y-axis in orange correspond to the norm of the gradients. Notice the "bump" in path-angle (close to $\alpha = 1$), characteristic of rotational dynamics.

of the Path-angle $c(\alpha)$ where the Jacobian is a constant matrix:

$$\boldsymbol{v}(\boldsymbol{\omega}) = \begin{bmatrix} \boldsymbol{S}_1 & \boldsymbol{B} \\ \boldsymbol{A} & \boldsymbol{S}_2 \end{bmatrix} (\boldsymbol{\omega} - \boldsymbol{\omega}^*) \quad \text{and thus} \quad \nabla \boldsymbol{v}(\boldsymbol{\omega}) = \begin{bmatrix} \boldsymbol{S}_1 & \boldsymbol{B} \\ \boldsymbol{A} & \boldsymbol{S}_2 \end{bmatrix} \quad \forall \boldsymbol{\omega} \,. \tag{4.2}$$

Depending on the choice of $\boldsymbol{S}_1, \boldsymbol{S}_2, \boldsymbol{A}$ and $\boldsymbol{B}$, we cover the following cases:

- $\boldsymbol{S}_1, \boldsymbol{S}_2 \succ 0, \boldsymbol{A} = \boldsymbol{B} = 0$: eigenvalues are real. Thm. 1 ensures that we only have *attraction*. Far from $\boldsymbol{\omega}^*$, the gradient points to $\boldsymbol{\omega}^*$ (See Fig. 6.2a) and thus $c(\alpha) = 1$ for $\alpha \ll 1$ and $c(\alpha) = -1$ for $\alpha \gg 1$. Since $\boldsymbol{\omega}'$ is not exactly $\boldsymbol{\omega}^*$, we observe a *quick sign switch* of the Path-angle around $\alpha = 1$. We plotted the average Path-angle over different approximate optima in Fig. 6.2a (see appendix for details).

- $\boldsymbol{S}_1, \boldsymbol{S}_2 = 0, \boldsymbol{A} = -\boldsymbol{B}^\top$: eigenvalues are pure imaginary. Thm. 1 ensures that we only have *rotations*. Far from the optimum the gradient is orthogonal to the direction that points to $\boldsymbol{\omega}$ (See Fig. 6.2b). Thus, $c(\alpha)$ vanishes for $\alpha \ll 1$ and $\alpha \gg 1$. Because $\boldsymbol{\omega}'$ is not exactly $\boldsymbol{\omega}^*$, around $\alpha = 1$, the gradient is tangent to the circles induced by the rotational dynamics and thus $c(\alpha) = \pm 1$. That is why in Fig. 6.2b we observe a *bump* in $c(\alpha)$ when α is close to 1.

- General high dimensional LSSP (3.4). The dynamics display both attraction and rotation. We observe a combination of the sign switch due to the attraction and the bump due to the rotation. The higher the bump, the closer we are to pure rotations. Since we are performing a low dimensional visualization, we actually project the gradient onto our direction of interest. That is why the Path-angle is significantly smaller than 1 in Fig. 6.2c.

5 Numerical results on GANs

Losses. We focus on two common GAN loss formulations: we consider both the original non-saturating GAN (NSGAN) formulation proposed in Goodfellow et al. [2014] and the WGAN-GP objective described in Gulrajani et al. [2017].

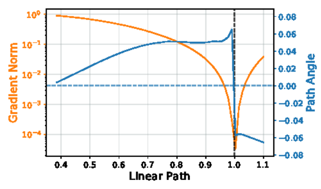
Datasets. We first propose to train a GAN on a toy task composed of a 1D mixture of 2 Gaussians (MoG) with 10,000 samples. For this task both the generator and discriminator are neural networks with 1 hidden layer and ReLU activations. We also train a GAN on MNIST, where we use the DCGAN architecture [Radford et al., 2016] with spectral normalization(see §3.2 for details). Finally we also look at the optimization landscape of a state of the art ResNet on CIFAR10 [Krizhevsky and Hinton, 2009].

Optimization methods. For the mixture of Gaussian (MoG) dataset, we used the full-batch extragradient method [Korpelevich, 1976, Gidel et al., 2019a]. We also tried to use standard batch gradient descent, but this led to unstable results indicating that gradient descent might indeed be unable to converge to stable stationary points due to the rotations (see §3.4). On MNIST and CIFAR10, we tested both Adam [Kingma and Ba, 2015] and ExtraAdam [Gidel et al., 2019a]. The observations made on models trained with both methods are very similar. ExtraAdam gives slightly better performance in terms of inception score [Salimans et al., 2016], and Adam sometimes converge to unstable points, thus we decided to only include the observations on ExtraAdam, for more details on the observations on Adam (see §3.5). As recommended by Heusel et al. [2017], we chose different learning rates for the discriminator and the generator. All the hyper-parameters and precise details about the experiments can be found in §3.1.

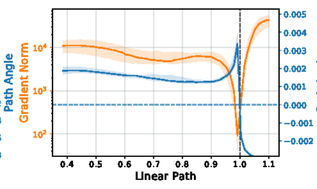
5.1 Evidence of rotation around locally stable stationary points in GANs

We first look, for all the different models and datasets, at the path-angles between a random initialization (initial point) and the set of parameters during training

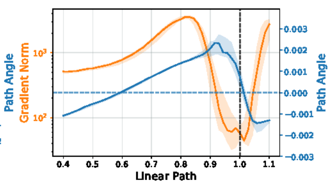
NSGAN



(a) MoG

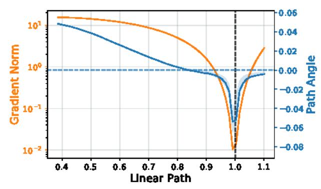


(b) MNIST, IS = 8.97

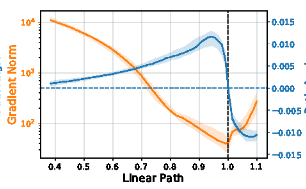


(c) CIFAR10, IS = 7.33

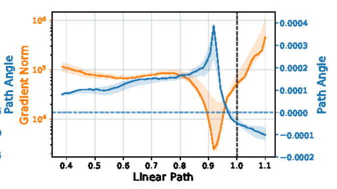
WGAN-GP



(d) MoG



(e) MNIST, IS = 9.46

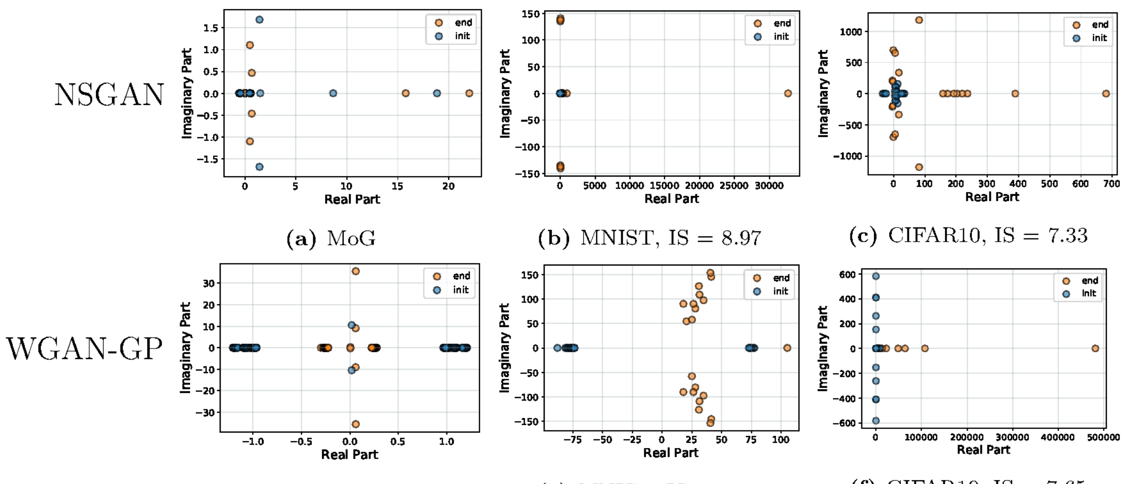


(f) CIFAR10, IS = 7.65

Figure 6.3: Path-angle for NSGAN (**top row**) and WGAN-GP (**bottom row**) trained on the different datasets, see Appendix 3.3 for details on how the path-angle is computed. For MoG the ending point is a generator which has learned the distribution. For MNIST and CIFAR10 we indicate the Inception score (IS) at the ending point of the interpolation. Notice the "bump" in path-angle (close to $\alpha = 1.0$), characteristic of games rotational dynamics, and absent in the minimization problem (d). Details on error bars in §3.3.

achieving the best performance (end point) (Fig. 6.3), and at the eigenvalues of the Jacobian of the game vector field for the same end point (Fig. 6.4). We're mostly interested in looking at the optimization landscape around LSSPs, so we first check if we are actually close to one. To do so we look at the gradient norm around the end point, this is shown by the orange curves in Fig.6.3, we can see that the norm of the gradient is quite small for all the models meaning that we are close to a stationary point. We also need to check that the point is stable, to do so we look at the eigenvalues of the Game in Fig. 6.4, if all the eigenvalues have positive real parts then the point is also stable. We observe that most of the time, the model has reached a LSSP. However we can see that this is not always the case, for example in Fig. 6.4d some of the eigenvalues have a negative real part. We still include those results since although the point is unstable it gives similar performance to a LSSP.

Our first observation is that all the GAN objectives on both datasets have a non zero rotational component. This can be seen by looking at the Path-angle in Fig. 6.3, where we always observe a bump, and this is also confirmed by the large imaginary part in the eigenvalues of the Jacobian in Fig. 6.4. The rotational component is clearly visible in Fig. 6.3d, where we see no sign switch and a clear bump similar to Fig. 6.2b. On MNIST and CIFAR10, with NSGAN and WGAN-GP (see Fig. 6.3), we observe a combination of a bump and a sign switch similar to Fig. 6.2c. Also Fig. 6.4 clearly shows the existence of imaginary eigenvalues with large magnitude. Fig. 6.4c and 6.4e. We can see that while almost all models exhibit rotations, the distribution of the eigenvalues are very different. In particular the complex



(a) MoG **(b)** MNIST, IS = 8.97 **(c)** CIFAR10, IS = 7.33

(d) MoG **(e)** MNIST, IS = 9.46 **(f)** CIFAR10, IS = 7.65

Figure 6.4: Eigenvalues of the Jacobian of the game for NSGAN (**top row**) and WGAN-GP (**bottom row**) trained on the different datasets. Large imaginary eigenvalues are characteristic of rotational behavior. Notice that NSGAN and WGAN-GP objectives lead to very different landscapes (see how the eigenvalues of WGAN-GP are shifted to the right of the imaginary axis). This could explain the difference in performance between NSGAN and WGAN-GP.

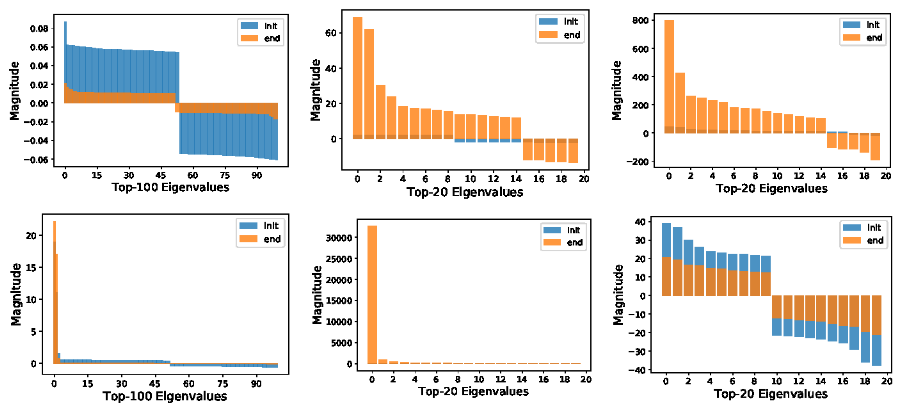
eigenvalues for NSGAN seems to be much more concentrated on the imaginary axis while WGAN-GP tends to spread the eigenvalues towards the right of the imaginary axis Fig. 6.4e. This shows that different GAN objectives can lead to very different landscapes, and has implications in terms of optimization, in particular that might explain why WGAN-GP performs slightly better than NSGAN.

5.2 The locally stable stationary points of GANs are not local Nash equilibria

As mentioned at the beginning of §5.1, the points we are considering are most of the times LSSP. To check if these points are also local Nash equilibria (LNE) we compute the eigenvalues of the Hessian of each player independently. If all the eigenvalues of each player are positive, it means that we have reached a DNE. Since the computation of the full spectrum of the Hessians is expensive, we restrict ourselves to the top-k eigenvalues with largest magnitude: exhibiting one significant negative eigenvalue is enough to indicate that the point considered is not in the neighborhood of a LNE. Results are shown in Fig. 6.5 and Fig. 6.6, from which we make several observations. First, we see that the generator never reaches a local minimum but instead finds a saddle point. This means that the algorithm converges to a LSSP which is not a LNE, while achieving good results with respect

Generator

Discriminator



(a) MoG **(b)** MNIST, IS = 8.97 **(c)** CIFAR10, IS = 7.33

Figure 6.5: NSGAN. Top k-Eigenvalues of the Hessian of each player (in terms of magnitude) in descending order. Top Eigenvalues indicate that the Generator does not reach a local minimum but a saddle point (for CIFAR10 actually both the generator and discriminator are at saddle points). Thus the training algorithms converge to LSSPs which are not Nash equilibria.

to our evaluation metrics. This raises the question whether convergence to a LNE is actually needed or if converging to a LSSP is sufficient to reach a good solution. We also observe a large difference in the eigenvalues of the discriminator when using the WGAN-GP v.s. the NSGAN objective. In particular, we find that the discriminator in NSGAN converges to a solution with very large positive eigenvalues compared to WGAN-GP. This shows that the discriminator in NSGAN converges to a much sharper minimum. This is consistent with the fact that the gradient penalty acts as a regularizer on the discriminator and prevents it from becoming too sharp.

6 Discussion

Across different GAN formulations, standard optimization methods and datasets, we consistently observed that GANs do not converge to local Nash equilibria. Instead the generator often ends up being at a saddle point of the generator loss function. However, in practice, these LSSP achieve really good generator performance metrics, which leads us to question whether we need a Nash equilibrium to get a generator with good performance in GANs and whether such DNE with good performance does actually exist. Moreover, we have provided evidence that the optimization landscapes of GANs typically have rotational components specific to games. We argue that these rotational components are part of the reason why GANs are challenging to train, in particular that the instabilities observed during training

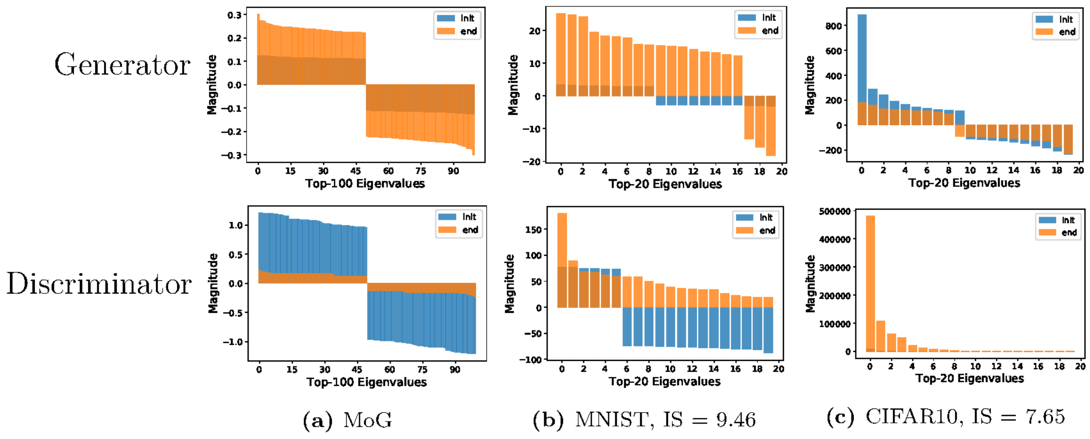


Figure 6.6: WGAN-GP. Top k-Eigenvalues of the Hessian of each player (in terms of magnitude) in descending order. Top Eigenvalues indicate that the Generator does not reach a local minimum but a saddle point. Thus the training algorithms converge to LSSPs which are not Nash equilibria.

may come from such rotations close to LSSP. It shows that simple low dimensional examples, such as for instance Dirac GAN, does capture some of the arising challenges for training large scale GANs, thus, motivating the practical use of method able to handle strong rotational components, such as extragradient [Gidel et al., 2019a], averaging [Yazıcı et al., 2019], optimism [Daskalakis et al., 2018] or gradient penalty based methods [Mescheder et al., 2017, Gulrajani et al., 2017].

7 Prologue to the Third Contribution

1 Article Details

Adversarial Example Games. *Avishek Joey Bose, Gauthier Gidel, Hugo Berard, Andre Cianflone, Pascal Vincent, Simon Lacoste-Julien and William L. Hamilton.* This paper was published at NeurIPS 2020 [Bose et al., 2020].

2 Contributions of the authors

Gauthier Gidel came up with the original idea for the game formulation and approached Hugo Berard and Joey Bose to work on the idea. The idea of the NoBox threat model, the architecture for the model and the experimental setup emerged from many discussions between all of them. They all participated to the writing of the paper, with Hugo Berard focusing more on the experiments by bringing his expertise in running large scale experiments and training adversarial games, Joey Bose focused more on the NoBox threat model contributing his experience with adversarial attacks, Gauthier Gidel focused on the theoretical results contributing his knowledge of game theory and coming up with the idea for the regularization. Pascal Vincent, Simon Lacoste-Julien and William L. Hamilton supervised the project.

3 Context and Impact

Adversarial attacks usually assume knowledge about the model they're trying to attack. In white-box attack, we usually assume that the attacker has full access to the model, by knowing the parameters of the model and being able to compute the gradient of the model. In the real world this is highly unrealistic, instead some work proposed to consider the case where the attacker only has access to the output of the model. However this is still unrealistic since most methods need to query the model several times to be able to craft the attack augmenting the chance of the attacker to be detected. Instead we proposed the NoBox attack model, which has only limited knowledge about the model it tries to attacks, and only requires to know the hypothesis class the model we want to attack belongs to. Several works proposed to study the transferability of adversarial attacks to other models, for example Papernot et al. [2016b] train a substitute model that is then used to craft attacks that are then used to attack the target model. Instead we propose a game called adversarial example game, that is specifically designed to generate attacks that transfer across all models in the same hypothesis class. We showed experimentally that using this formulation we can generate attacks that transfer across a wide variety of models and outperform the existing approach in the NoBox setting.

Limitations and Remarks. Although we do not assume access to the original training set to train the attacker, we still assume that the distribution of samples used to train the attacker are close to the real distribution. In practice we only tested on different splits of the same dataset. Further experiments would be needed to test if the attacker can be trained with examples that come from a different distribution and still be able to transfer attacks. AEG trains the generator and the representative classifier jointly, according to Prop. 1 AEG admits a Nash-Equilibrium, at this equilibrium both the generator and representative classifier are optimal, thus we should expect the representative classifier to be robust, however we haven't tested this hypothesis and leave this for future work to see if this game can also be used to train more robust classifiers.

8 Adversarial Example Games

Abstract

The existence of adversarial examples capable of fooling trained neural network classifiers calls for a much better understanding of possible attacks to guide the development of safeguards against them. This includes attack methods in the challenging *non-interactive blackbox* setting, where adversarial attacks are generated without any access, including queries, to the target model. Prior attacks in this setting have relied mainly on algorithmic innovations derived from empirical observations (e.g., that momentum helps), lacking principled transferability guarantees. In this work, we provide a theoretical foundation for crafting transferable adversarial examples to entire hypothesis classes. We introduce *Adversarial Example Games* (AEG), a framework that models the crafting of adversarial examples as a min-max game between a generator of attacks and a classifier. AEG provides a new way to design adversarial examples by adversarially training a generator and a classifier from a given hypothesis class (e.g., architecture). We prove that this game has an equilibrium, and that the optimal generator is able to craft adversarial examples that can attack any classifier from the corresponding hypothesis class. We demonstrate the efficacy of AEG on the MNIST and CIFAR-10 datasets, outperforming prior state-of-the-art approaches with an average relative improvement of 29.9% and 47.2% against undefended and robust models (Table 8.2 & 8.3) respectively.

1 Introduction

Adversarial attacks on deep neural nets expose critical vulnerabilities in traditional machine learning systems [Moosavi-Dezfooli et al., 2016, Athalye et al., 2018, Sun et al., 2018, Bose and Aarabi, 2018]. In order to develop models that are robust to such attacks, it is imperative that we improve our theoretical understanding of different attack strategies. While there has been considerable progress in understanding the theoretical underpinnings of adversarial attacks in relatively permissive settings (e.g. *whitebox* adversaries; Madry et al. [2018]), there remains a substantial gap between theory and practice in more demanding and realistic threat models.

In this work, we provide a theoretical framework for understanding and analyzing adversarial attacks in the highly-challenging ***No****n-interactive black****Box** adversary* (NoBox) setting, where the attacker has no direct access, including input-output queries, to the target classifier it seeks to fool. Instead, the attacker must generate attacks by optimizing against some representative classifiers, which are assumed to come from a similar hypothesis class as the target.

The NoBox setting is a much more challenging setting than more traditional threat models, yet it is representative of many real-world attack scenarios, where the attacker cannot interact with the target model [Carlini et al., 2019]. Indeed, this setting—as well as the general notion of transferring attacks between classifiers—has generated an increasing amount of empirical interest [Dong et al., 2019, Liu et al., 2016, Xie et al., 2019, Wu et al., 2020]. The field, however, currently lacks the necessary theoretical foundations to understand the feasibility of such attacks.

Contributions. To address this theoretical gap, we cast NoBox attacks as a kind of *adversarial example game* (AEG). In this game, an attacker generates adversarial examples to fool a representative classifier from a given hypothesis class, while the classifier itself is trained to detect the correct labels from the adversarially generated examples. Our first main result shows that the Nash equilibrium of an AEG leads to a distribution of adversarial examples effective against *any* classifier from the given function class. More formally, this adversarial distribution is guaranteed to be the most effective distribution for attacking the hardest-to-fool classifiers within the hypothesis class, providing a worst-case guarantee for attack success against an arbitrary target. We further show that this optimal adversarial distribution admits a natural interpretation as being the distribution that maximizes a form of restricted conditional entropy over the target dataset, and we provide detailed analysis on simple parametric models to illustrate the characteristics of this optimal adversarial distribution. Note that while AEGs are latent games [Gidel et al., 2020], they are distinct from the popular generative adversarial networks (GANs) [Goodfellow et al., 2014]. In AEGs, there is *no* discrimination task between two datasets (generated one and real one); instead, there is a standard supervised (multi-class) classification task on an adversarial dataset.

Guided by our theoretical results we instantiate AEGs using parametric functions —i.e. neural networks, for both the attack generator and representative classifier and show the game dynamics progressively lead to a stronger attacker and robust classifier pairs. We empirically validate AEG on standard CIFAR and MNIST benchmarks and achieve state-of-the-art performance —compared to existing heuristic approaches— in nearly all experimental settings (e.g., transferring attacks to unseen architectures and attacking robustified models), while also maintaining a firm theoretical grounding.

2 Background and Preliminaries

Suppose we are given a classifier $f : \mathcal{X} \to \mathcal{Y}$, an input datapoint $x \in \mathcal{X}$, and a class label $y \in \mathcal{Y}$, where $f(x) = y$. The goal of an adversarial attack is to produce an adversarial example $x' \in \mathcal{X}$, such that $f(x') \neq y$, and where the distance[1] $d(x, x') \leq \epsilon$. Intuitively, the attacker seeks to fool the classifier f into making the wrong prediction on a point x', which is ϵ-close to a real data example x.

Adversarial attacks and optimality. A popular setting in previous research is to focus on generating *optimal* attacks on a single classifier f [Carlini and Wagner, 2017a, Madry et al., 2018]. Given a loss function ℓ, used to evaluate f, an adversarial attack is said to be optimal if,

$$x' \in \arg\max_{x' \in \mathcal{X}} \ell(f(x'), y)\,, \quad \text{s.t.} \quad d(x, x') \leq \epsilon\,. \tag{2.1}$$

In practice, attack strategies that aim to realize (2.1) optimize adversarial examples x' directly using the gradient of f. In this work, however, we consider the more general setting of generating attacks that are optimal against entire hypothesis classes $\mathcal{F}$, a notion that we formalize below.

2.1 NoBox Attacks

Threat models specify the formal assumptions of an attack (e.g., the information the attacker is assumed to have access to), which is a core aspect of adversarial attacks. For example, in the popular *whitebox* threat model, the attacker is assumed to have full access to the model f's parameters and outputs [Szegedy et al., 2014, Goodfellow et al., 2015a, Madry et al., 2018]. In contrast, the *blackbox* threat model assumes restricted access to the model, e.g., only access to a limited number of input-out queries [Chen et al., 2017, Ilyas et al., 2017, Papernot et al., 2016a]. Overall, while they consider different access to the target model, traditional whitebox and blackbox attacks both attempt to generate adversarial examples that are optimal for a specific target (i.e., Equation 2.1).

[1]We assume that the ℓ_∞ is used in this work, Goodfellow et al. [2015a], Madry et al. [2018] , but our results generalize to any distance d.

In this paper, we consider the more challenging setting of ***no**n-interactive black**Box** (**NoBox**)* attacks, intending to generate successful attacks against an unknown target. In the NoBox setting, we assume no interactive access to a target model; instead, we only assume access to a target dataset and knowledge of the function class to which a target model belongs. Specifically, the NoBox threat model relies on the following key definitions:

- **The target model** f_t. The adversarial goal is to attack some target model $f_t : \mathcal{X} \to \mathcal{Y}$, which belongs to an hypothesis class $\mathcal{F}$. Critically, the adversary has *no access* to f_t *at any time*. Thus, in order to attack f_t, the adversary must develop attacks that are effective against the entirety of $\mathcal{F}$.
- **The target examples** $\mathcal{D}$. The dataset $\mathcal{D}$ contains the examples (x, y) that attacker seeks to corrupt.
- **An hypothesis class** $\mathcal{F}$. As noted above, we assume that the attacker has access to a hypothesis class $\mathcal{F}$ to which the target model f_t belongs.[2] One can incorporate in $\mathcal{F}$ as much prior knowledge one has on f_t (e.g., the architecture, dataset, training method, or regularization), going from exact knowledge of the target $\mathcal{F} = \{f_t\}$ to almost no knowledge at all (e.g., $\mathcal{F} = \{f \in \text{DenseNets}\}$).
- **A reference dataset** $\mathcal{D}_{\textbf{ref}}$. The reference dataset $\mathcal{D}_{\text{ref}}$, which is *similar* to the training data of the target model (e.g., sampled from the same distribution) is used to reduce the size of the hypothesis class $\mathcal{F}$ (e.g., we know that the target model perfoms well at classification on $\mathcal{D}_{\text{ref}}$).
- **A representative classifier** f_c. Finally, we assume that the attacker has the ability to optimize a representative classifier f_c from the hypothesis class $\mathcal{F}$.

Given these four key components, we formalize the NoBox setting as follows:

Definition 1. *The NoBox threat model corresponds to the setting where the attacker (i) knows a hypothesis class $\mathcal{F}$ that the target model f_t belongs to, (ii) has access to a reference dataset $\mathcal{D}_{ref}$ that is similar to the the dataset used to train f_t (e.g., sampled from the same distribution), and (iii) can optimize a representative classifier $f_c \in \mathcal{F}$. The attacker has no other knowledge of—or access to—the target model f_t (e.g., no queries to f_t are allowed). The goal is, for the attacker, to use this limited knowledge to corrupt the examples in a given target dataset $\mathcal{D}$.*

Our definition of a NoBox adversary (Def. 1) formalizes similar notions used in previous work (e.g., see Def. 3 in Tramèr et al. [2018]). Previous work also often refers to related settings as generating *blackbox transfer* attacks, since the goal is to

[2]Previous work [Tramèr et al., 2018] usually assumes to have access to the architecture of f_t; we are more general by assuming access to a hypothesis class $\mathcal{F}$ containing f_t; e.g., DenseNets can represent ConvNets.

attack the target model f_t while only having access to a representative classifier f_c [Dong et al., 2019, Liu et al., 2016, Xie et al., 2019].

Note, that our assumptions regarding dataset access are relatively weak. Like prior work, the attacker is given the target data (i.e., the examples to corrupt) as input, but this is constitutive of the task (i.e., we need access to a target example in order to corrupt it). Our only assumption is to have access to a reference dataset $\mathcal{D}_{\text{ref}}$, which is *similar* to the dataset used to train the target model. We do not assume access to the exact training set. A stronger version of this assumption is made in prior works on blackbox transfer, as these approaches must craft their attacks on a known source model which is pretrained on the same dataset as the target model [Tramèr et al., 2018].

3 Adversarial Example Games

In order to understand the theoretical feasibility of NoBox attacks, we view the attack generation task as a form of *adversarial game*. The players are the *generator* network g—which learns a conditional distribution over adversarial examples—and the representative classifier f_c. The goal of the generator network is to learn a conditional distribution of adversarial examples, which can fool the representative classifier f_c. The representative classifier f_c, on the other hand, is optimized to detect the true label y from the adversarial examples (x', y) generated by g. A critical insight in this framework is that the generator and the representative classifier are *jointly* optimized in a maximin game, making the generator's adversarial distribution at the equilibrium theoretically effective against *any* classifier from the hypothesis class $\mathcal{F}$ that f_c is optimized over. At the same time, we will see in Proposition 1 that the min and max in our formulation (AEG) can be switched. It implies that, while optimized, the model f_c converges to a *robust classifier* against any attack generated by the generator g [Madry et al., 2018, Wald, 1945], leading to increasingly powerful attacks as the adversarial game progresses.

Framework. Given an input-output pair of target datapoints $(x, y) \sim \mathcal{D}$, the generator network g is trained to learn a distribution of adversarial examples $p_{\text{cond}}(\cdot|x, y)$ that—conditioned on an example to attack (x, y)—maps a prior distribution p_z on $\mathcal{Z}$ onto a distribution on $\mathcal{X}$. The classifier network f_c is simultaneously optimized to perform robust classification over the resulting distribution p_g defined in (3.1) (below). Overall, the generator g and the classifier f_c play the following, two-player zero-sum game:

$$\max_{g\in\mathcal{G}_\epsilon}\min_{f_c\in\mathcal{F}} \mathbb{E}_{(x,y)\sim\mathcal{D},z\sim p_z}[\ell(f_c(g(x,y,z)),y)] =: \varphi(f_c,g), \tag{AEG}$$

where the generator $g \in \mathcal{G}_\epsilon$ is restricted by the similarity constraint $d(g(x,y,z),x) \leq \epsilon\,, \forall x,y,z \in \mathcal{X}\times\mathcal{Y}\times\mathcal{Z}$. Once the generator g is trained, one can generate adversarial examples against any classifier in $f_t \in \mathcal{F}$, without queries, by simply sampling $z \sim p_z$ and computing $g(x,y,z)$.

Connection with NoBox attacks. The NoBox threat model (Def. 1) corresponds to a setting where the attacker does not know the target model f_t but only a hypothesis class $\mathcal{F}$ such that $f_t \in \mathcal{F}$. With such knowledge, one cannot hope to be better than the *most pessimistic situation* where f_t is the best defender in $\mathcal{F}$. Our maximin formulation (AEG) encapsulates such a worst-case scenario, where the generator aims at finding attacks against the best performing f in $\mathcal{F}$.

Objective of the generator. When trying to attack infinite capacity classifiers—i.e., $\mathcal{F}$ contains any measurable function—the goal of the generator can be seen as generating the adversarial distribution p_g with the highest expected conditional entropy $\mathbb{E}_x[\sum_y p_g(y|x)\log p_g(y|x)]$, where p_g is defined as

$$(x',y)\sim p_g \Leftrightarrow x' = g(x,y,z)\,,\ (x,y)\sim\mathcal{D}\,,\ z\sim p_z \quad\text{with}\quad d(x',x)\leq\epsilon\,. \tag{3.1}$$

When trying to attack a specific hypothesis class $\mathcal{F}$ (e.g., a particular CNN architecture), the generator aims at maximizing a notion of restricted entropy defined implicitly through the class $\mathcal{F}$. Thus, the optimal generator in an (AEG) is primarily determined by the statistics of the target dataset $\mathcal{D}$ itself, rather any specifics of a target model. We formalize these high level concepts in §4.2.

Regularizing the Game. In practice, the target f_t is usually trained on a non-adversarial dataset and performs well at a standard classification task. In order to reduce the size of the class $\mathcal{F}$, one can bias the representative classifier f_c towards performing well on a standard classification task with respect to $\mathcal{D}_{\text{ref}}$, which leads to the following game:

$$\max_{g\in\mathcal{G}_\epsilon}\min_{f_c\in\mathcal{F}} \mathbb{E}_{(x,y)\sim\mathcal{D},z\sim p_z}[\ell(f_c(g(x,y,z)),y)] + \lambda\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{ref}}}[\ell(f_c(x),y)] =: \varphi_\lambda(f,g). \tag{3.2}$$

Note that $\lambda = 0$ recovers (AEG). Such modifications in the maximin objective as well as setting the way the models are trained (e.g., optimizer, regularization, additional dataset) biases the training of the f_c and corresponds to an implicit incorporation of prior knowledge on the target f_t in the hypothesis class $\mathcal{F}$. We note that in practice, using a non-zero value for λ is essential to achieve the most effective attacks as the prior knowledge acts as a regularizer that incentivizes g to craft attacks against classifiers that behave well on data similar to $\mathcal{D}_{\text{ref}}$.

4 Theoretical results

When playing an adversarial example game, the generator and the representative classifier try to beat each other by maximizing their own objective. In games, a standard notion of optimality is the concept of Nash equilibrium [Nash, 1951] where each player cannot improve its objective value by unilaterally changing its strategy. The minimax result in Prop. 1 implies the existence of a Nash equilibrium for the game, consequently providing a well defined target for learning (we want to learn the equilibrium of that game). Moreover, a Nash equilibrium is a stationary point for gradient descent-ascent dynamics; we can thus hope for achieving such a solution by using a gradient-descent-ascent-based learning algorithm on (AEG).[3]

Proposition 1. *If ℓ is convex (e.g., cross entropy or mean squared loss), the distance $x \mapsto d(x, x')$ is convex for any $x' \in \mathcal{X}$, one has access to any measurable g respecting the proximity constraint in (3.1), and the hypothesis class $\mathcal{F}$ is convex, then we can switch min and max in (AEG), i.e.,*

$$\min_{f_c \in \mathcal{F}} \max_{g \in \mathcal{G}_\epsilon} \varphi_\lambda(f_c, g) = \max_{g \in \mathcal{G}_\epsilon} \min_{f_c \in \mathcal{F}} \varphi_\lambda(f_c, g) \tag{4.1}$$

Proof sketch. We first notice that, by (3.1) any g corresponds to a distribution p_g and thus we have,

$$\varphi(f_c, g) := \mathbb{E}_{(x,y)\sim\mathcal{D}, z\sim p_z}[\ell(f_c(g(x, y, z)), y)] = \mathbb{E}_{(x',y)\sim p_g}[\ell(f_c(x'), y)] =: \varphi(f_c, p_g)$$

Consequently, we also have $\varphi_\lambda(f_c, g) = \varphi_\lambda(f_c, p_g)$. By noting $\Delta_\epsilon := \{p_g \,:\, g \in \mathcal{G}_\epsilon\}$, we have that,

$$\min_{f_c \in \mathcal{F}} \max_{p_g \in \Delta_\epsilon} \varphi_\lambda(f_c, p_g) = \min_{f_c \in \mathcal{F}} \max_{g \in \mathcal{G}_\epsilon} \varphi_\lambda(f_c, g) \quad \text{and} \quad \max_{p_g \in \Delta_\epsilon} \min_{f_c \in \mathcal{F}} \varphi_\lambda(f_c, p_g) = \max_{g \in \mathcal{G}_\epsilon} \min_{f_c \in \mathcal{F}} \varphi_\lambda(f_c, g)$$

In other words, we can replace the optimization over the generator $g \in \mathcal{G}_\epsilon$ with an optimization over the set of possible adversarial distributions Δ_ϵ induced by any $g \in \mathcal{G}_\epsilon$. This equivalence holds by the construction of Δ_ϵ, which ensures that $\max_{g \in \mathcal{G}_\epsilon} \varphi_\lambda(f_c, g) = \max_{p_g \in \Delta_\epsilon} \varphi_\lambda(f_c, p_g)$ for any $f_c \in \mathbb{F}$.

We finally use Fan's theorem [Fan, 1953] after showing that $(f_c, p_g) \mapsto \varphi_\lambda(f_c, p_g)$ is convex-concave (by convexity of ℓ and linearity of $p_g \mapsto \mathbb{E}_{p_g}$) and that Δ_ϵ is a compact convex set. In particular, Δ_ϵ is compact convex under the assumption that we can achieve any measurable g (detailed in §C). $\square$

[3]Note that, similarly as in practical GANs training, when the classifier and the generator are parametrized by neural networks, providing convergence guarantees for a gradient based method in such a nonconvex-nonconcave minimax game is an open question that is outside of the scope of this work.

The convexity assumption on the hypothesis class $\mathcal{F}$, Prop. 1 applies in two main cases of interest: (i) infinite capacity, i.e., when $\mathcal{F}$ is any measurable function. (ii) linear classifiers with *fixed* features $\psi : \mathcal{X} \to \mathbb{R}^p$, i.e., $\mathcal{F} = \{w^\top \psi(\cdot)\,,\, w \in \mathbb{R}^{|\mathcal{Y}| \times p}\}$. This second setting is particularly useful to build intuitions on the properties of (AEG), as we will see in §4.1 and Fig. 8.1. The assumption that we have access to any measurable g, while relatively strong, is standard in the literature and is often stated in prior works as "if g has enough capacity" [Goodfellow et al., 2015a, Prop. 2]. Even if the class of neural networks with a fixed architecture do not verify the assumption of this proposition, the key idea is that neural networks are good candidates to approximate that equilibrium because they are universal approximators [Hornik, 1991] and they form a set that is "almost convex" [Gidel et al., 2020]. Proving a similar minimax theorem by only considering neural networks is a challenging problem that has been considered by Gidel et al. [2020] in a related setting. It requires a fined grained analysis of the property of certain neural network architecture and is only valid for approximate minimax. We believe such considerations outside of the scope of this work.

4.1 A simple setup: binary classification with logistic regression

Let us now consider a binary classification setup where $\mathcal{Y} = \{\pm 1\}$ and $\mathcal{F}$ is the class of linear classifiers with linear features, i.e $f_w(x) = w^\top x$. In this case, the payoff of the game (AEG) is,

$$\varphi(f_\omega, g) := \mathbb{E}_{(x,y)\sim\mathcal{D},\, z\sim p_z}[\log(1 + e^{-y\cdot w^\top g(x,y,z)})] \tag{4.2}$$

This example is similar to the one presented in [Goodfellow et al., 2015a]. However, our purpose is different since we focus on characterizing the optimal generator in (4.1). We show that the optimal generator can attack any classifier in $\mathcal{F}$ by shifting the means of the two classes of the dataset $\mathcal{D}$.

Proposition 2. *If the generator is allowed to generate any ℓ_∞ perturbations. The optimal linear representative classifier is the solution of the following ℓ_1 regularized logistic regression*

$$w^* \in \arg\min_w \mathbb{E}_{(x,y)\sim\mathcal{D}}[\log(1 + e^{-y\cdot w^\top x + \epsilon\|w\|_1})]\,. \tag{4.3}$$

Moreover if ω^* has no zero entry, the optimal generator is $g^*(x,y) = x - y\cdot\epsilon\,\mathrm{sign}(w^*)$, is deterministic *and the pair (f_{w^*}, g^*) is a Nash equilibrium of the game (4.2).*

A surprising fact is that, unlike in the general setting of Prop. 1, the generator in

Prop.2 is deterministic (i.e., does not depend on a latent variable z).[4] This follows from the simple structure of classifiers in this class, which allow for a closed form solution for g^*. In general, one cannot expect to achieve an equilibrium with a deterministic generator. Indeed, with this example, our goal is simply to illustrate how the optimal generator can attack an entire class of functions with limited capacity: linear classifiers are mostly sensitive to the mean of the distribution of each class; the optimal generator exploits this fact by moving these means closer to the decision boundary.

4.2 General multi-class classification

In this section, we show that, for a given hypothesis class $\mathcal{F}$, the generated distribution achieving the global maximin against $f_c \in \mathcal{F}$ can be interpreted as the distribution with the highest $\mathcal{F}$-entropy. For a given distribution p_g, its $\mathcal{F}$-entropy is the minimum expected risk under p_g one can achieve in $\mathcal{F}$.

Definition 2. *For a given distribution $(x, y) \sim p_g$ we define the $\mathcal{F}$-entropy of p_g as*

$$H_{\mathcal{F}}(p_g) := \min_{f_c \in \mathcal{F}} \mathbb{E}_{(x,y)\sim p_g}[\ell(f_c(x), y)] \qquad \textit{where } \ell \textit{ is the cross entropy loss.} \tag{4.4}$$

Thus $\mathcal{F}$-entropy quantifies the amount of “classification information" available in p_g using the class of classifiers $\mathcal{F}$. If the $\mathcal{F}$-entropy is large, $(x, y) \sim p_g$ cannot be easily classified with a function f_c in $\mathcal{F}$. Moreover, it is an upper-bound on the *expected conditional entropy* of the distribution p_g.

Proposition 3. *The $\mathcal{F}$-entropy is a decreasing function of $\mathcal{F}$, i.e., for any $\mathcal{F}_1 \subset \mathcal{F}_2$,*

$$H_{\mathcal{F}_1}(p_g) \geq H_{\mathcal{F}_2}(p_g) \geq H_y(p_g) := \mathbb{E}_{x\sim p_x}[H(p_g(\cdot|x))] \,.$$

where $H(p(\cdot|x)) := \sum_{y\in\mathcal{Y}} p(y|x) \ln p(y|x)$ is the entropy of the conditional distribution $p(y|x)$.

Here p_g is defined as in (3.1) and implicity depends on $\mathcal{D}$. For a given class $\mathcal{F}$, the solution to an (AEG) game can be seen as one which finds a regularized adversarial distribution of maximal $\mathcal{F}$-entropy,

$$\max_{g\in\mathcal{G}_\epsilon} \min_{f_c\in\mathcal{F}} \varphi_\lambda(f_c, g) = (1 + \lambda) \max_{g\in\mathcal{G}_\epsilon} H_{\mathcal{F}}\left(\tfrac{1}{(1+\lambda)} p_g + \tfrac{\lambda}{(1+\lambda)} \mathcal{D}_{\text{ref}}\right)] \,, \tag{4.5}$$

[4]Note also that one can generalize Prop. 2 to a perturbation with respect to a general norm $\|\cdot\|$, in that case, the ϵ-regularization for the classifier would be with respect to the dual norm $\|\cdot\|_* := \max_{\|u\|\leq 1}\langle\cdot, u\rangle$. E.g., as previously noted by Goodfellow et al. [2015a], ℓ_∞ adversarial perturbation leads to a ℓ_1 regularization.

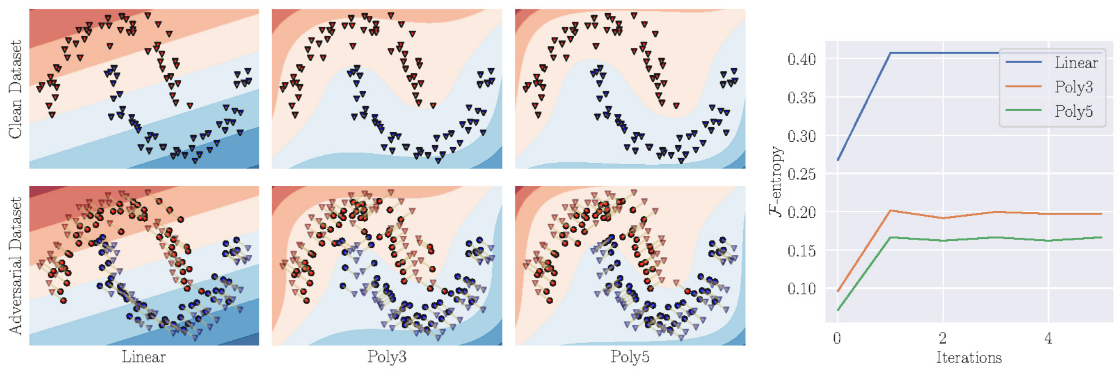


Figure 8.1: Illustration of Proposition 3 for three classes of classifiers in the context of logistic regression for the two moon dataset of scikit-learn [Pedregosa et al., 2011] with linear and polynomial (of degree 3 and 5) features. **Left:** Scatter plot of the clean or adversarial dataset and the associated optimal decision boundary. For the adversarial dataset, each corresponding clean example is represented with a ▲/▲ and is connected to its respective adversarial example ●/●. **Right:** value of the $\mathcal{F}$-entropy for the different classes as a function of the number of iterations.

where the distribution $\frac{1}{(1+\lambda)}p_g + \frac{\lambda}{(1+\lambda)}\mathcal{D}_{\text{ref}}$ is the mixture of the generated distribution p_g and the empirical distribution over the dataset $\mathcal{D}_{\text{ref}}$. This alternative perspective on the game (AEG) shares similarities with the divergence minimization perspective on GANs [Huang et al., 2017a]. However, while in GANs it represents a divergence between two distributions, in (AEG) this corresponds to a notion of entropy.

A high-level interpretation of $\mathcal{F}$-entropy maximization is that it implicitly defines a metric for distributions which are challenging to classify with only access to classifiers in $\mathcal{F}$. Overall, the optimal generated distribution p_g can be seen as the most adversarial dataset against the class $\mathcal{F}$.

Properties of the $\mathcal{F}$-entropy. We illustrate the idea that the optimal generator and the $\mathcal{F}$-entropy depend on the hypothesis class $\mathcal{F}$ using a simple example. To do so, we perform logistic regression (4.2) with linear and polynomial (of degree 3 and 5) features (respectively called Linear, Poly3, and Poly5) on the two moon dataset of scikit-learn [Pedregosa et al., 2011]. Note that we have Linear $\subset$ Poly3 $\subset$ Poly5. For simplicity, we consider a deterministic generator $g(x, y)$ that is realized by computing the maximization step via 2D grid-search on the ϵ neighborhood of x. We train our models by successively fully solving the minimization step and the maximization step in (4.2).

We present the results in Figure 8.1. One iteration corresponds to the computation of the optimal classifier against the current adversarial distribution p_g (also giving the value of the $\mathcal{F}$-entropy), followed by the computation of the new optimal adversarial

p'_g against this new classifier. The left plot illustrates the fact that the way of attacking a dataset depends on the class considered. For instance, when considering linear classifiers, the attack is a uniform translation on all the data-points of the same class. While when considering polynomial features, the optimal adversarial dataset pushes the the corners of the two moons closer together. In the right plot, we can see an illustration of Proposition 3, where the $\mathcal{F}$-entropy takes on a smaller value for larger classes of classifiers.

5 Attacking in the Wild: Experiments and Results

We investigate the application of our AEG framework to produce adversarial examples against MNIST and CIFAR-10 classifiers. First we investigate our performance in a challenging NoBox setting where we must attack an unseen target model with knowledge of only its hypothesis class (i.e., architecture) and a sample of similar training data (§5.1). Following this, we investigate how well AEG attacks transfer across architectures (§5.2), as well as AEG's performance attacking robust classifiers (§5.3).

Experimental setup. We perform all attacks, including baselines, with respect to the ℓ_∞ norm constraint with $\epsilon = 0.3$ for MNIST and $\epsilon = 0.03125$ for CIFAR-10. For AEG models, we train both generator (g) and representative classifier (f_c) using stochastic gradient descent-ascent with the ExtraAdam optimizer [Gidel et al., 2019a] and held out target models, f_t, are trained offline using SGD with Armijo line search [Vaswani et al., 2019]. Full details of our model architectures, including hyperparameters, employed in our AEG framework can be found in Appendix §4.[5]

Baselines. Throughout our experiments we rely on four standard blackbox transfert attack strategies adapted to the NoBox setting: the Momentum-Iterative Attack (MI-Attack) [Dong et al., 2018], the Input Diversity (DI-Attack) [Xie et al., 2019], the Translation-Invariant (TID-Attack) [Dong et al., 2019] and the Skip Gradient Method (SGM-Attack) [Wu et al., 2020]. For fair comparison, we inherit all hyperparameter settings from their respective papers. Note that SGM-attack is only defined with architectures that contain skip connections (e.g. ResNets).

AEG Architecture. The high-level architecture of our AEG framework is illustrated in Figure 8.2. The generator takes the input x and encode it into $\psi(x)$, then the generator uses this encoding to compute a probability vector $p(\psi(x))$ in

[5]Code: `https://github.com/joeybose/Adversarial-Example-Games.git`

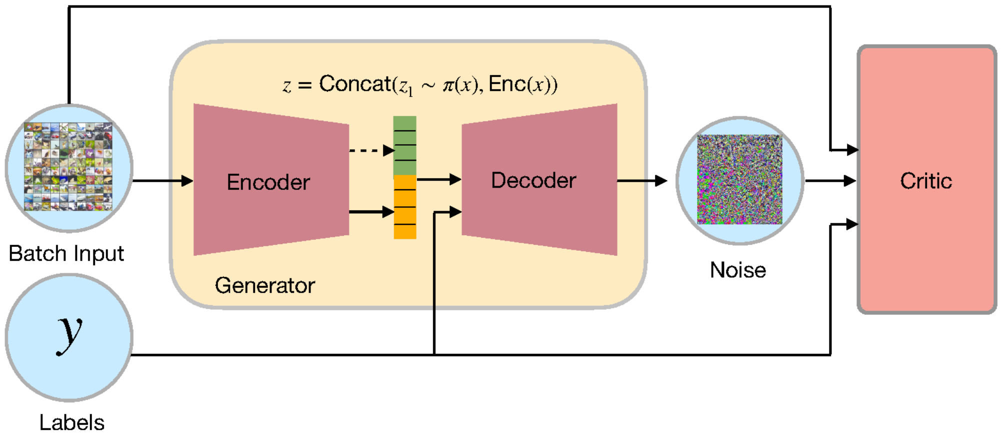


Figure 8.2: AEG framework architecture

the probability simplex of size K, the number of classes. Using this probability vector, the network then samples a categorical variable z according to a multinomial distribution of parameter $p(\psi(x))$. Intuitively, this category may correspond to a target for the attack. The gradient is backprogated across this categorical variable using the gumble-softmax trick Jang et al. [2017], Maddison et al. [2017]. Finally, the decoder takes as input $\psi(x)$, z and the label y to output an adversarial perturbation δ such that $\|\delta\| \leq \epsilon$. In order to generate adversarial perturbations over images that obey ϵ-ball constraints, we employ a scaled tanh output layer to scale the output of the generator to $(0, 1)$, subtract the clean images, and finally apply an elementwise multiplication by ϵ. We then compute $\ell(f(x + \delta), y)$ where f is the critic and ℓ the cross entropy loss. Further details can be found in Appendix §4.

5.1 NoBox Attacks on a Known Architecture Class but Unknown Train Set

We first evaluate the AEG framework in a NoBox setting, where we know only the architecture of the target model and have only access to a sample of similar training data (but not the exact training data of the target model). To simulate having access to a similar (but not identical dataset) as the target model, for each dataset we create random equally-sized splits of the data (10000 examples per splits). Within each split we use one fold to train the split classifier which acts as the representative classifiers for all attackers who are then evaluated their ability to fool the remaining split classifiers on unseen target examples $\mathcal{D}$. For the MNIST dataset we consider LeNet classifier [LeCun et al., 2015b], while for CIFAR-10 we consider ResNet-18 [He et al., 2016]. Table 8.1 shows the results of our experiments on this task, averaged across all splits and folds. We see that our AEG approach achieves state-of-the-art results, either outperforming or matching (within a 95% confidence interval) all baselines in both settings. Note that this task is significantly more challenging than many prior blackbox attack setups, which assume access to the full training data of the target model.[6]

Dataset	MI-Attack	DI-Attack	TID-Attack	SGM-Attack	AEG (Ours)
MNIST	87.5 ± 2.7	**89.5 ± 2.5**	85.4 ± 2.8 †	N/A	**89.5 ± 3.2**
CIFAR-10	56.8 ± 1.2 †	84.0 ± 1.5 †	9.1 ± 1.6 †	60.5 ± 1.5 †	**87.0 ± 2.1**

Table 8.1: Attack success rates, averaged across target models with 95% confidence intervals shown. †indicates a statistically significant result as determined by the paired T-test when compared to AEG. CIFAR-10 results are with a Res18 architecture.

5.2 NoBox Attacks Across Distinct Architectures

We now consider NoBox attacks where we do not know the architecture of the target model but where the training data is known—a setting previously referred to as blackbox transfer [Tramèr et al., 2018]. For evaluation, we use CIFAR-10 and train 10 instances of VGG-16 [Simonyan and Zisserman, 2015], ResNet-18 (RN-18) [He et al., 2016], Wide ResNet (WR) [Zagoruyko and Komodakis, 2016], DenseNet-121 (DN-121) [Huang et al., 2017b] and Inception-V3 architectures (Inc-V3) [Szegedy et al., 2016]. Here, we optimize the attack approaches against a single pre-trained classifier from a particular architecture and then evaluate their attack success on classifiers from distinct architectures averaged over 5 instantiations. Our findings when using ResNet-18, DenseNet-121 and the VGG-16 as the source architecture are provided in Table 2. Overall we find that AEG beats all other approaches and lead to a new state of the art. In particular AEG outperforms the best baseline in each setting by an average of 29.9% across the different source architectures with individual average gains of 9.4%, 36.2%, and 44.0% when using a RN-18 model, DN-121, and VGG-16 source models respectively.

5.3 NoBox Attacks Against Robust Classifiers

We now test the ability of our AEG framework to attack target models that have been robustified using adversarial and ensemble adversarial training [Madry et al.,

[6]We include results on a more permissive settings with access to the full training data in Appendix 3.1

Source	Attack	VGG-16	RN-18	WR	DN-121	Inc-V3
	Clean	11.2 ± 1.8	13.1 ± 4.0	6.8 ± 1.4	11.2 ± 2.8	9.9 ± 2.6
RN-18	MI-Attack	63.9 ± 2.6	74.6 ± 0.8	63.1 ± 2.4	72.5 ± 2.6	67.9 ± 3.2
	DI-Attack	77.4 ± 3.4	90.2 ± 1.6	74.0 ± 2.0	87.1 ± 2.6	**85.8 ± 1.6**
	TID-Attack	21.6 ± 2.6	26.5 ± 4.8	14.0 ± 3.0	22.3 ± 3.2	19.8 ± 1.8
	SGM-Attack	68.4 ± 3.6	79.5 ± 1.0	64.3 ± 3.2	73.8 ± 2.0	70.6 ± 3.4
	AEG (Ours)	**93.8 ± 0.7**	**97.1 ± 0.4**	**80.2 ± 2.2**	**93.1 ± 1.3**	**88.4 ± 1.6**
DN-121	MI-Attack	54.3 ± 2.2	62.5 ± 1.8	56.3 ± 2.6	66.1 ± 3.0	65.0 ± 2.6
	DI-Attack	61.1 ± 3.8	69.1 ± 1.6	61.9 ± 2.2	77.1 ± 2.4	71.6 ± 3.2
	TID-Attack	21.7 ± 2.4	23.8 ± 3.0	14.0 ± 2.8	21.7 ± 2.2	19.3 ± 2.4
	SGM-Attack	51.6 ± 1.4	60.2 ± 2.6	52.6 ± 1.8	64.7 ± 3.2	61.4 ± 2.6
	AEG (Ours)	**93.7 ± 1.0**	**97.3 ± 0.6**	**81.8 ± 3.0**	**96.7 ± 0.8**	**92.7 ± 1.6**
VGG-16	MI-Attack	49.9 ± 0.2	50.0 ± 0.4	46.7 ± 0.8	50.4 ± 1.2	50.0 ± 0.6
	DI-Attack	65.1 ± 0.2	64.5 ± 0.4	58.8± 1.2	64.1 ± 0.6	60.9 ± 1.2
	TID-Attack	26.2 ± 1.2	24.0 ± 1.2	13.0 ± 0.4	20.8 ± 1.4	18.8 ± 0.4
	AEG (Ours)	**97.5 ± 0.4**	**96.1 ± 0.5**	**85.2 ± 2.2**	**94.1 ± 1.2**	**89.5 ± 1.3**

Table 8.2: Error rates on $\mathcal{D}$ for average NoBox architecture transfer attacks with $\epsilon = 0.03125$. The ± correspond to 2 standard deviations (95.5% confidence interval for normal distributions).

2018, Tramèr et al., 2018]. For evaluation against PGD adversarial training, we use the public models as part of the MNIST and CIFAR-10 adversarial examples challenge.[7] For ensemble adversarial training, we follow the approach of Tramèr et al. [2018] (see Appendix 4.3). We report our results in Table 3 and average the result of stochastic attacks over 5 runs. We find that AEG achieves state-of-the-art performance in all settings, proving an average improvement in success rates of 54.1% across all robustified MNIST models and 40.3% on robustified CIFAR-10 models.

6 Related Work

In addition to non-interactive blackbox adversaries we compare against, there exists multiple hybrid approaches that combine crafting attacks on surrogate models which then serve as a good initialization point for queries to the target model

[7] `https://github.com/MadryLab/[x]_challenge`, for `[x]` in {`cifar10`,`mnist`}. Note that our threat model is more challenging than these challenges as we use non-robust source models.

Dataset	Defence	Clean	MI-Att[†]	DI-Att	TID-Att	SGM-Att [†]	AEG (Ours)
MNIST	A_{ens4}	0.8	43.4	42.7	16.0	N/A	**65.0**
	B_{ens4}	0.7	20.7	22.8	8.5	N/A	**50.0**
	C_{ens4}	0.8	73.8	30.0	9.5	N/A	**80.0**
	D_{ens4}	1.8	84.4	76.0	81.3	N/A	**86.7**
	Madry-Adv	0.8	2.0	3.1	2.5	N/A	**5.9**
CIFAR-10	RN-18_{ens3}	16.8	17.6	21.6	33.1	19.9	**52.2**
	WR_{ens3}	12.8	18.4	20.6	28.8	18.0	**49.9**
	DN-121_{ens3}	21.5	20.3	22.7	31.3	21.9	**41.4**
	Inc-$V3_{ens3}$	14.8	19.5	42.2*	30.2*	35.5*	**47.5**
	Madry-Adv	12.9	17.2	16.6	16.6	16.0	**21.6**

Table 8.3: Error rates on $\mathcal{D}$ for NoBox known architecture attacks against Adversarial Training and Ensemble Adversarial Training. * Attacks were done using WR. [†] Deterministic attack.

Papernot et al. [2016a], Shi et al. [2019], Huang and Zhang [2020]. Other notable approaches to craft blackbox transfer attacks learning ghost networks Li et al. [2018c], transforming whitebox gradients with small ResNets Li et al. [2020], and transferability properties of linear classifiers and 2-layer ReLu Networks Charles et al. [2019]. There is also a burgeoning literature of using parametric models to craft adversarial attacks such as the Adversarial Transformation Networks framework and its variants Baluja and Fischer [2018], Xiao et al. [2018]. Similar in spirit to our approach many attack strategies benefit from employing a latent space to craft attacks Zhao et al. [2018], Tu et al. [2019], Bose et al. [2019]. However, unlike our work, these strategies cannot be used to attack entire hypothesis classes.

Adversarial prediction games between a learner and a data generator have also been studied in the literature [Brückner et al., 2012], and in certain situations correspond to a Stackelberg game Brückner and Scheffer [2011]. While similar in spirit, our theoretical framework is tailored towards crafting adversarial attacks against a fixed held out target model in the novel NoBox threat model and is a fundamentally different attack paradigm. Finally, Erraqabi et al. [2018] also investigate an adversarial game framework as a means for building robust representations in which an additional discriminator is trained to discriminate adversarial example from natural ones, based on the representation of the current classifier.

7 Conclusion

In this paper, we introduce the Adversarial Example Games (AEG) framework which provides a principled foundation for crafting adversarial attacks in the NoBox threat model. Our work sheds light on the existence of adversarial examples as a natural consequence of restricted entropy maximization under a hypothesis class and leads to an actionable strategy for attacking all functions taken from this class. Empirically, we observe that our approach leads to state-of-the-art results when generating attacks on MNIST and CIFAR-10 in a number of challenging NoBox attack settings. Our framework and results point to a promising new direction for theoretically-motivated adversarial frameworks. However, one major challenge is scaling up the AEG framework to larger datasets (e.g., ImageNet), which would involve addressing some of the inherent challenges of saddle point optimization [Berard et al., 2020]. Investigating the utility of the AEG framework for training robustified models is another natural direction for future work.

Broader Impact

Adversarial attacks, especially ones under more realistic threat models, pose several important security, ethical, and privacy risks. In this work, we introduce the NoBox attack setting, which generalizes many other blackbox transfer settings, and we provide a novel framework to ground and study attacks theoretically and their transferability to other functions within a class of functions. As the NoBox threat model represents a more realistic setting for adversarial attacks, our research has the potential to be used against a class of machine learning models in the wild. In particular, in terms of risk, malicious actors could use approaches based on our framework to generate attack vectors that compromise production ML systems or potentially bias them toward specific outcomes.

As a concrete example, one can consider creating transferrable examples in the physical world, such as the computer vision systems of autonomous cars. While prior works have shown the possibility of such adversarial examples —i.e., adversarial traffic signs, we note that there is a significant gap in translating synthetic adversarial examples to adversarial examples that reside in the physical world [45]. Understanding and analyzing the NoBox transferability of adversarial examples to the physical world—in order to provide public and academic visibility on these risks—is an important direction for future research. Based on the known risks of designing new kinds of adversarial attacks—discussed above—we now outline the ways in which our research is informed by the intent to mitigate these potential societal risks. For instance, our research demonstrates that one can successfully craft adversarial attacks even in the challenging NoBox setting. It raises many important considerations when developing robustness approaches. A straightforward extension is to consider our adversarial example game (AEG) framework as a tool for training robust models. On the theoretical side, exploring formal verification of neural networks against NoBox adversaries is an exciting direction for continued exploration. As an application, ML practitioners in the industry may choose to employ new forms of A/B testing with different types of adversarial examples, of which AEG is one method to robustify and stress test production systems further. Such an application falls in line with other general approaches to red teaming AI systems [10] and verifiability in AI development. In essence, the goal of such approaches, including adversarial examples for robustness, is to align AI systems' failure modes to those found in human decision making.

Acknowledgments and Disclosure of Funding

The authors would like to acknowledge Olivier Mastropietro, Chongli Qin and David Balduzzi for helpful discussions as well as Sebastian Lachapelle, Pouya Bashivan, Yanshuai Cao, Gavin Ding, Ioannis Mitliagkas, Nadeem Ward, and Damien Scieur for reviewing early drafts of this work.

Funding. This work is partially supported by the Canada CIFAR AI Chair Program (held at Mila), NSERC Discovery Grant RGPIN-2019-05123 (held by Will Hamilton at McGill), NSERC Discovery Grant RGPIN-2017-06936, an IVADO Fundamental Research Project grant PRF-2019-3583139727, and a Google Focused Research award (both held at U. Montreal by Simon Lacoste-Julien). Joey Bose was also supported by an IVADO PhD fellowship, Gauthier Gidel by a Borealis AI fellowship and by the Canada Excellence Research Chair in "Data Science for Real-Time Decision-making" (held at Polytechnique by Andrea Lodi), and Andre Cianflone by a NSERC scholarship and a Borealis AI fellowship. Simon Lacoste-Julien and Pascal Vincent are CIFAR Associate Fellows in the Learning in Machines & Brains program. Finally, we thank Facebook for access to computational resources.

Competing interests. Joey Bose was formerly at FaceShield.ai which was acquired in 2020. W.L. Hamilton was formerly a Visiting Researcher at Facebook AI Research. Simon Lacoste-Julien additionally works part time as the head of the SAIT AI Lab, Montreal from Samsung.

9 Conclusions, Discussions, and Perspectives

1 Summary and Conclusions

In this thesis we looked at the challenges and applications of adversarial games in machine learning. We first looked at the challenges of optimizing such games in the context of training GANs, looking at instabilities resulting from the adversarial nature of the game, and proposing new algorithms with better properties. We then proposed a new application of games to the generation of adversarial attacks that transfer across models.

In the first contribution, we bridged the gap between the GAN and the optimization literature by casting GANs as an instance of a variational inequality problem. This enabled us to leverage tools from the variational inequality literature to provide algorithms with convergence guarantees in the monotone settings for training GANs. In particular we proposed to use extrapolation and averaging and combined them with Adam, we called the resulting algorithm ExtraAdam and showed through several experiments on GANs that it provided better results. We also proposed another variant that uses extrapolation form the past, that only requires to compute one gradient at each iteration and prove its convergence for stochastic monotone VIP.

In the second contribution, we proposed to experimentally analyze the vector field of GANs. To do so we proposed a new tool called Path-angle to analyze whether a stationary point exhibits rotational or attractive behavior. By running several experiments with different GANs architectures on different datasets, we showed that most GAN problems converge to local stationary point that have a rotational behavior. We also looked at the eigenvalues of the Jacobian of the vector field of the game, and showed that GANs do not converge to Nash-equilibria but instead converge to local stable stationary points.

In the third and final contribution, we proposed a new game that we call adversarial example game to train a generator to produce adversarial attacks that transfer across models. In this game, an attacker tries to generate adversarial examples that fools a representative classifier that belongs to a given hypothesis class, while the classifier itself is trained to correctly classify the perturbed examples. We first showed that such games admit a Nash equilibrium, and that at this equilibrium the adversarial examples produced by the generator are effective against any classifier in the hypothesis class. We then tested AEG on several datasets and architectures, and showed that it achieved state-of-the-art performance outperforming previous approaches in almost all experiments.

2 Discussions and Perspectives

Using game formulations in machine learning can lead to new algorithms and new advances, as demonstrated by the success of GANs. However using game formulations comes with several challenges as we have highlighted in this thesis. In particular games which are adversarial in nature often have rotational dynamics which affects convergence and performance. Better understanding those dynamics and having better algorithms could open the door to more applications of games in machine learning.

Equilibrium in Games. As mentioned in the second contribution of this thesis, GANs generally do not converge to a Nash equilibrium which raises the question of their existence. Farnia and Ozdaglar [2020] showed that indeed such equilibria might not exist and instead propose a new notion of equilibria called proximal equilibrium. Other notions of optimality have also been proposed, Jin et al. [2020] propose to take into account the sequential nature of the game and proposed the notion of local minimax equilibrium. Finally, Fiez et al. [2020] look at the notion of Stackelberg equilibrium and analyze the convergence of GDA to such equilibrium.

Non-Monotone Games. Another challenge of games, is that machine learning models are often highly non-linear leading to problems which are non-monotone and thus requires new assumptions and convergence guarantees. Several recent works have tried to extend existing results to non-monotone games. For example, Lin et al. [2020] provides convergence guarantees for GDA on nonconvex-concave games. Yang et al. [2020] that derive convergence guarantees for SGDA on a class of nonconvex-nonconcave games. Gorbunov et al. [2022] provide a general analysis of extragradient for a class of non-monotone games called quasi-strongly-monotone VIP.

Hamiltonian Gradient Methods. Another avenue consists in designing new algorithms with better convergence guarantees for games. Mescheder et al. [2017] proposed Hamiltonian gradient descent and consensus optimization to avoid the rotational behavior in games. Loizou et al. [2020] provided convergence guarantees for HGD for a class of nonconvex-nonconcave game, and provided an extension with linear convergence using variance reduction. An analysis of both consensus optimization and SGDA was provided in Loizou et al. [2021]. Azizian et al. [2020] showed that HGD is optimal for bilinear games and that CO can be accelerated.

GANs. Recently GANs have had great success, paving the way to high quality and high resolution image generation. Brock et al. [2019] were the first to generate images at 512x512 resolution with impressive results. Recently, Sauer et al. [2022] achieved event better quality and higher resolution, producing images at a 1024x1024 resolution. However Brock et al. [2019] mentions instabilities during training where the generator collapses if trained for too long, showing the need for a better understanding of such dynamics and more stable algorithms. Recently, other models have also been proposed that achieve results of similar quality to Sauer et al. [2022]. For example, Dhariwal and Nichol [2021] use diffusion models to achieve state-of-the-art results.

Games in ML. Finally, games have a large potential for being applied to a wide variety of ML problems. For example we can mention, Ahuja et al. [2020] who propose invariant risk minimization games for domain generalization, Rahme et al. [2020] who formulate Auction Design as a two-player game between two neural networks, and multi-agent reinforcement learning [Zhang et al., 2021] who is becoming increasingly popular can be formulated as a game between several agents, and can heavily benefit from advances in both game theory and machine learning.

Bibliography

L. Adolphs, H. Daneshmand, A. Lucchi, and T. Hofmann. Local saddle point optimization: A curvature exploitation approach. *arXiv preprint arXiv:1805.05751*, 2018.

K. Ahuja, K. Shanmugam, K. Varshney, and A. Dhurandhar. Invariant risk minimization games. *arXiv preprint arXiv:2002.04692*, 2020.

N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

G. Alain, N. Le Roux, and P.-A. Manzagol. Negative eigenvalues of the hessian in deep neural networks. *arXiv*, 2019.

Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein. Square attack: a query-efficient black-box adversarial attack via random search. *Sixteenth European Conference On Computer Vision (ECCV)*, 2020.

M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Hk4_qw5xe.

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

A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. In *The Thirty-fifth International Conference on Machine Learning (ICML)*, 2018.

K. E. Atkinson. *An introduction to numerical analysis*. John Wiley & Sons, 2003.

W. Azizian, D. Scieur, I. Mitliagkas, S. Lacoste-Julien, and G. Gidel. Accelerating smooth games by manipulating spectral shapes. In *AISTATS*, 2020.

D. Balduzzi, S. Racaniere, J. Martens, J. Foerster, K. Tuyls, and T. Graepel. The mechanics of n-player differentiable games. In *ICML*, 2018.

S. Baluja and I. Fischer. Learning to attack: Adversarial transformation networks. In *Thirty-second aaai conference on artificial intelligence (AAAI)*, 2018.

J. Bardeen and W. H. Brattain. The transistor, a semi-conductor triode. *Physical Review*, 74(2):230, 1948.

H. Berard, G. Gidel, A. Almahairi, P. Vincent, and S. Lacoste-Julien. A closer look at the optimization landscapes of generative adversarial networks. In *ICLR*, 2020.

A. Beznosikov, E. Gorbunov, H. Berard, and N. Loizou. Stochastic gradient descent-ascent: Unified theory and new efficient methods. *arXiv preprint arXiv:2202.07262*, 2022.

S. Bhambri, S. Muku, A. Tulasi, and A. B. Buduru. A survey of black-box adversarial attacks on computer vision models. *arXiv preprint arXiv:1912.01667*, 2019.

P. Billingsley. *Convergence of probability measures*. John Wiley & Sons, 1999.

A. J. Bose and P. Aarabi. Adversarial attacks on face detectors using neural net based constrained optimization. In *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2018.

A. J. Bose, A. Cianflone, and W. Hamiltion. Generalizable adversarial attacks using generative models. *arXiv preprint arXiv:1905.10864*, 2019.

A. J. Bose, G. Gidel, H. Berrard, A. Cianflone, P. Vincent, S. Lacoste-Julien, and W. L. Hamilton. Adversarial example games. *arXiv preprint arXiv:2007.00720*, 2020.

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

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

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

M. Brückner and T. Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011.

M. Brückner, C. Kanzow, and T. Scheffer. Static prediction games for adversarial learning problems. *The Journal of Machine Learning Research*, 13(1):2617–2654, 2012.

S. Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

J. Buckman, A. Roy, C. Raffel, and I. Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations (ICLR 2018)*, 2018.

N. Carlini and D. Wagner. Magnet and efficient defenses against adversarial attacks are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017a.

N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017b.

N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

Z. Charles, H. Rosenberg, and D. Papailiopoulos. A geometric perspective on the transferability of adversarial directions. In *The Twenty-Second International Conference on Artificial Intelligence and Statistics*, 2019.

T. Chavdarova, G. Gidel, F. Fleuret, and S. Lacoste-Julien. Reducing noise in gan training with variance reduced extragradient. In *NeurIPS*, pages 393–403, 2019.

G. H. Chen and R. T. Rockafellar. Convergence rates in forward–backward splitting. *SIAM Journal on Optimization*, 1997.

P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the tenth ACM Workshop on Artificial Intelligence and Security*. ACM, 2017.

C.-K. Chiang, T. Yang, C.-J. Lee, M. Mahdavi, C.-J. Lu, R. Jin, and S. Zhu. Online optimization with gradual variations. In *COLT*, pages 6–1, 2012.

A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *AISTATS*, pages 192–204, 2015.

G. P. Crespi, A. Guerraggio, and M. Rocca. Minty variational inequality and optimization: Scalar and vector case. In A. Eberhard, N. Hadjisavvas, and D. T. Luc, editors, *Generalized Convexity, Generalized Monotonicity and Applications*, 2005.

F. Croce and M. Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. *Eighth International Conference on Learning Representations (ICLR)*, 2019.

F. Croce and M. Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *Thirty-seventh International Conference on Machine Learning (ICML)*, 2020.

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

Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NeurIPS*, 2014.

A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar. Stochastic activation pruning for robust adversarial defense. *Sixth International Conference on Learning Representations (ICLR)*, 2018.

G. W. Ding, L. Wang, and X. Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

G. W. Ding, Y. Sharma, K. Y. C. Lui, and R. Huang. MMA training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*, 2020.

Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

Y. Dong, T. Pang, H. Su, and J. Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht. Essentially no barriers in neural network energy landscape. In *ICML*, 2018.

J. Du, H. Zhang, J. T. Zhou, Y. Yang, and J. Feng. Query-efficient meta attack to deep neural networks. *Eighth International Conference on Learning Representations (ICLR 2020)*, 2020.

A. d'Aspremont, D. Scieur, A. Taylor, et al. Acceleration methods. *Foundations and Trends® in Optimization*, 5(1-2):1–245, 2021.

A. Erraqabi, A. Baratin, Y. Bengio, and S. Lacoste-Julien. A3t: Adversarially augmented adversarial training. *arXiv preprint arXiv:1801.04055*, 2018.

F. Facchinei and J.-S. Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer.

K. Fan. Minimax theorems. *Proceedings of the National Academy of Sciences of the United States of America*, 1953.

F. Farnia and A. Ozdaglar. Do gans always have nash equilibria? In *International Conference on Machine Learning*, pages 3029–3039. PMLR, 2020.

W. Fedus, M. Rosca, B. Lakshminarayanan, A. M. Dai, S. Mohamed, and I. Goodfellow. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. In *ICLR*, 2018.

T. Fiez, B. Chasnov, and L. 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.

G. Gidel, T. Jebara, and S. Lacoste-Julien. Frank-Wolfe algorithms for saddle point problems. *AISTATS*, 2017.

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

G. Gidel, R. A. Hemmat, M. Pezeshki, R. Lepriol, G. Huang, S. Lacoste-Julien, and I. Mitliagkas. Negative momentum for improved game dynamics. In *AISTATS*, 2019b.

G. Gidel, D. Balduzzi, W. M. Czarnecki, M. Garnelo, and Y. Bachrach. Minimax theorem for latent games or: How i learned to stop worrying about mixed-nash and love neural nets. *arXiv preprint arXiv:2002.05820*, 2020.

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

Z. Gong, W. Wang, and W.-S. Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1705.04960*, 2017.

I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv:1701.00160*, 2016.

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

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *Third International Conference of Learning Representations (ICLR)*, 2015a.

I. J. Goodfellow, O. Vinyals, and A. M. Saxe. Qualitatively characterizing neural network optimization problems. In *ICLR*, 2015b.

E. Gorbunov, H. Berard, G. Gidel, and N. Loizou. Stochastic extragradient: General analysis and improved rates. In *International Conference on Artificial Intelligence and Statistics*, pages 7865–7901. PMLR, 2022.

R. M. Gower. Convergence theorems for gradient descent. *Lecture notes for Statistical Optimization*, 2018.

R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, and P. Richtárik. Sgd: General analysis and improved rates. In *International Conference on Machine Learning*, pages 5200–5209. PMLR, 2019.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

P. Grnarova, K. Y. Levy, A. Lucchi, T. Hofmann, and A. Krause. An online learning approach to generative adversarial networks. In *ICLR*, 2018.

K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein GANs. In *NeurIPS*, 2017.

C. Guo, M. Rana, M. Cisse, and L. Van Der Maaten. Countering adversarial images using input transformations. *Sixth International Conference on Learning Representations (ICLR)*, 2018.

C. Guo, J. R. Gardner, Y. You, A. G. Wilson, and K. Q. Weinberger. Simple black-box adversarial attacks. In *Thirthy-Sixth International Conference on Machine Learning (ICML)*, 2019.

T. S. Guzella and W. M. Caminhas. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222, 2009.

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

E. Hazan. Lecture notes: Optimization for machine learning. *arXiv preprint arXiv:1909.03550*, 2019.

E. Hazan, K. Singh, and C. Zhang. Efficient regret minimization in non-convex games. In *ICML*, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

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

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 1991.

K. Hornik, M. Stinchcombe, H. White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

G. Huang, H. Berard, A. Touati, G. Gidel, P. Vincent, and S. Lacoste-Julien. Parametric adversarial divergences are good task losses for generative modeling. *arXiv preprint arXiv:1708.02511*, 2017a.

G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017b.

Z. Huang and T. Zhang. Black-box adversarial attack with transferable model-based embedding. *Eighth International Conference on Learning Representations (ICLR)*, 2020.

A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Query-efficient black-box adversarial examples. *arXiv preprint arXiv:1712.07113*, 2017.

A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box adversarial attacks with limited queries and information. *Thirty-fifth International Conference on Machine Learning (ICML)*, 2018.

A. Ilyas, L. Engstrom, and A. Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *Seventh International Conference on Learning Representations (ICLR)*, 2019.

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

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *Fifth International Conference on Learning Representations (ICLR)*, 2017.

S. Jelassi, A. Mensch, G. Gidel, and Y. Li. Adam is no better than normalized sgd: Dissecting how adaptivity improves gan performance. *preprint*, 2021.

L. Jiang, X. Ma, S. Chen, J. Bailey, and Y.-G. Jiang. Black-box adversarial attacks on video recognition models. In *Proceedings of the twenty-seventh ACM International Conference on Multimedia*, 2019.

C. Jin, P. Netrapalli, and M. Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *International conference on machine learning*, pages 4880–4889. PMLR, 2020.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

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

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

L. V. Kantorovich. Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422, 1960.

T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

K. Kawaguchi. Deep learning without poor local minima. In *NeurIPS*, pages 586–594, 2016.

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

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

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

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

T. Larsson and M. Patriksson. A class of gap functions for variational inequalities. *Math. Program.*, 1994.

Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database, 2010.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015a.

Y. LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 2015b.

C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

A. M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes; par AM Legendre...* chez Firmin Didot, libraire pour lew mathematiques, la marine, l . . . , 1806.

H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. In *NeurIPS*, 2018a.

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

Y. Li, A. Schwing, K.-C. Wang, and R. Zemel. Dualing GANs. In *NeurIPS*, 2017.

Y. Li, S. Bai, Y. Zhou, C. Xie, Z. Zhang, and A. Yuille. Learning transferable adversarial examples via ghost networks. *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2018c.

Y. Li, L. Li, L. Wang, T. Zhang, and B. Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. *Thirty-Sixth International Conference on Machine Learning (ICML)*, 2019.

Y. Li, S. Bai, C. Xie, Z. Liao, X. Shen, and A. L. Yuille. Regional homogeneity: Towards learning transferable universal adversarial perturbations against defenses. *European Conference on Computer Vision (ECCV)*, 2020.

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

T. Lin, C. Jin, and M. Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *International Conference on Machine Learning*, pages 6083–6093. PMLR, 2020.

Y. Liu, X. Chen, C. Liu, and D. Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

N. Loizou, H. Berard, A. Jolicoeur-Martineau, P. Vincent, S. Lacoste-Julien, and I. Mitliagkas. Stochastic hamiltonian gradient methods for smooth games. In *International Conference on Machine Learning*, pages 6370–6381. PMLR, 2020.

N. Loizou, H. Berard, G. Gidel, I. Mitliagkas, and S. Lacoste-Julien. Stochastic gradient descent-ascent and consensus optimization for smooth games: Convergence analysis under expected co-coercivity. *Advances in Neural Information Processing Systems*, 34:19095–19108, 2021.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *Fifth International Conference on Learning Representations (ICLR)*, 2017.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

E. Mazumdar, L. J. Ratliff, and S. S. Sastry. On gradient-based learning in continuous games. *SIAM Journal on Mathematics of Data Science*, 2(1):103–131, 2020.

E. V. Mazumdar, M. I. Jordan, and S. S. Sastry. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *arXiv preprint arXiv:1901.00838*, 2019.

P. Mertikopoulos, B. Lecouat, H. Zenati, C.-S. Foo, V. Chandrasekhar, and G. Piliouras. Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile. In *ICLR*, 2019.

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

L. Mescheder, A. Geiger, and S. Nowozin. Which Training Methods for GANs do actually Converge? In *ICML*, 2018.

L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. In *ICLR*, 2017.

J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. *Sixth International Conference on Learning Representations (ICLR)*, 2017.

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

A. Mladenovic, J. Bose, H. berard, W. L. Hamilton, S. Lacoste-Julien, P. Vincent, and G. Gidel. Online adversarial attacks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=bYGSzbCM_i.

S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

A. Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

V. Nagarajan and J. Z. Kolter. Gradient descent GAN optimization is locally stable. In *NeurIPS*, 2017.

J. Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

J. F. Nash et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 1950.

A. Nedić and A. Ozdaglar. Subgradient methods for saddle-point problems. *J Optim Theory Appl*, 2009.

A. 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 J. Optim.*, 2004.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 2009.

Y. Nesterov. *Introductory Lectures On Convex Optimization.* Springer, 1983.

Y. Nesterov. Dual extrapolation and its applications to solving variational inequalities and related problems. *Math. Program.*, 2007.

Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical programming*, 140(1):125–161, 2013.

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

S. Nowozin, B. Cseke, and R. Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *NeurIPS*, 2016.

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

N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016a.

N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 1(2):3, 2016b.

N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security.* ACM, 2017.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

W. Paul. Electromagnetic traps for charged and neutral particles. *Reviews of modern physics*, 1990.

B. A. Pearlmutter. Fast exact multiplication by the hessian. *Neural computation*, 6 (1):147–160, 1994.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011.

B. T. Polyak. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653, 1963.

L. D. Popov. A modification of the arrow-hurwicz method for search of saddle points. *Mathematical notes of the Academy of Sciences of the USSR*, 1980.

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

J. Rahme, S. Jelassi, and S. M. Weinberg. Auction learning as a two-player game. In *International Conference on Learning Representations*, 2020.

A. Rakhlin and K. Sridharan. Online learning with predictable sequences. In *COLT*, 2013.

A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

L. J. Ratliff, S. A. Burden, and S. S. Sastry. On the characterization of local nash equilibria in continuous games. In *IEEE Transactions on Automatic Control*, 2016.

A. Rényi et al. On measures of entropy and information. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, volume 1. Berkeley, California, USA, 1961.

H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

L. Sagun, L. Bottou, and Y. LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv*, 2016.

L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv*, 2017.

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

A. Sauer, K. Schwarz, and A. Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

J. Schmidt, M. R. Marques, S. Botti, and M. A. Marques. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 5(1):1–36, 2019.

Y. Shi, S. Wang, and Y. Han. Curls & whey: Boosting black-box adversarial attacks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587): 484–489, 2016.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *Third International Conference on Learning Representations (ICLR)*, 2015.

Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *Sixth International Conference on Learning Representations (ICLR)*, 2018.

L. Sun, M. Tan, and Z. Zhou. A survey of practical adversarial example attacks. *Cybersecurity*, 1(1):9, 2018.

I. Sutskever. *Training recurrent neural networks*. University of Toronto Toronto, Canada, 2013.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *Second International Conference on Learning Representations (ICLR)*, 2014.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

R. Thompson, T. Harmon, and M. Ball. The rotating-saddle trap: A mechanical analogy to rf-electric-quadrupole ion trapping? *Canadian journal of physics*, 2002.

F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses. *Sixth International Conference on Learning Representations (ICLR)*, 2018.

P. Tseng. On linear convergence of iterative methods for the variational inequality problem. *Journal of Computational and Applied Mathematics*, 60(1-2):237–252, 1995.

C.-C. Tu, P. Ting, P.-Y. Chen, S. Liu, H. Zhang, J. Yi, C.-J. Hsieh, and S.-M. Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

S. Vaswani, A. Mishkin, I. Laradji, M. Schmidt, G. Gidel, and S. Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *NeurIPS*, 2019.

F. Verhulst. *Nonlinear differential equations and dynamical systems.* Springer Science & Business Media, 1989.

C. Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

A. Wald. Statistical decision functions which minimize the maximum risk. *Annals of Mathematics*, 1945.

B. Wu and A. Kumar. Extreme ultraviolet lithography: A review. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena*, 25(6):1743–1761, 2007.

D. Wu, Y. Wang, S.-T. Xia, J. Bailey, and X. Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. *Eigth International Conference on Learning Representations (ICLR)*, 2020.

C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song. Generating adversarial examples with adversarial networks. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

H. Xu, Y. Ma, H. Liu, D. Deb, H. Liu, J. Tang, and A. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *arXiv preprint arXiv:1909.08072*, 2019.

A. Yadav, S. Shah, Z. Xu, D. Jacobs, and T. Goldstein. Stabilizing adversarial nets with prediction methods. In *ICLR*, 2018.

J. Yang, N. Kiyavash, and N. He. Global convergence and variance-reduced optimization for a class of nonconvex-nonconcave minimax problems. *arXiv preprint arXiv:2002.09621*, 2020.

Y. Yazıcı, C.-S. Foo, S. Winkler, K.-H. Yap, G. Piliouras, and V. Chandrasekhar. The unusual effectiveness of averaging in gan training. In *ICLR*, 2019.

F. Yousefian, A. Nedić, and U. V. Shanbhag. Optimal robust smoothing extragradient algorithms for stochastic variational inequality problems. In *CDC*. IEEE, 2014.

S. Zagoruyko and N. Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, September 2016.

H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

K. Zhang, Z. Yang, and T. Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021.

Z. Zhao, D. Dua, and S. Singh. Generating natural adversarial examples. *Sixth International Conference on Learning Representations*, 2018.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017.

C. L. Zitnick, L. Chanussot, A. Das, S. Goyal, J. Heras-Domingo, C. Ho, W. Hu, T. Lavril, A. Palizhati, M. Riviere, et al. An introduction to electrocatalyst design using machine learning for renewable energy storage. *arXiv preprint arXiv:2010.09435*, 2020.

A A Variational Inequality Perspective on Generative Adversarial Networks

1 Definitions

In this section we recall usual definitions and lemmas from convex analysis. We start with the definitions and lemmas regarding the projection mapping.

1.1 Projection mapping

Definition 3. *The projection P_Ω onto Ω is defined as,*

$$P_\Omega(\boldsymbol{\omega}') \in \operatorname*{arg\,min}_{\boldsymbol{\omega}'\in\Omega} \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 \,. \tag{1.1}$$

When Ω is a convex set this projection is unique. This is a consequence of the following lemma that we will use in the following sections: the *non-expansiveness* of the projection onto a convex set.

Lemma 1. *Let Ω a convex set, the projection mapping $P_\Omega : \mathbb{R}^d \to \Omega$ is nonexpansive, i.e.,*

$$\|P_\Omega(\boldsymbol{\omega}) - P_\Omega(\boldsymbol{\omega}')\|_2 \leq \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2 \,, \quad \forall \boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega \,. \tag{1.2}$$

This is standard convex analysis result which can be found for instance in [Boyd and Vandenberghe, 2004]. The following Lemma is also standard in convex analysis and its proof uses similar arguments as the proof of Lemma 1.

Lemma 2. *Let $\boldsymbol{\omega} \in \Omega$ and $\boldsymbol{\omega}^+ := P_\Omega(\boldsymbol{\omega} + \boldsymbol{u})$ then for all $\boldsymbol{\omega}' \in \Omega$ we have,*

$$\|\boldsymbol{\omega}^+ - \boldsymbol{\omega}'\|_2^2 \leq \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 + 2\boldsymbol{u}^\top(\boldsymbol{\omega}^+ - \boldsymbol{\omega}') - \|\boldsymbol{\omega}^+ - \boldsymbol{\omega}\|_2^2 \,. \tag{1.3}$$

Proof of Lemma 2. We start by simply developing,

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

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

1.2 Smoothness and Monotonicity of the operator

Another important property used is the Lipschitzness of an operator.

Definition 4. *A mapping $F : \mathbb{R}^p \to \mathbb{R}^d$ is said to be L-Lipschitz if,*

$$\|F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}')\|_2 \leq L \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2\,, \quad \forall \boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega\,. \tag{1.4}$$

Definition 5. *A differentiable function $f : \Omega \to \mathbb{R}$ is said to be μ-*strongly convex *if*

$$f(\boldsymbol{\omega}) \geq f(\boldsymbol{\omega}') + \nabla f(\boldsymbol{\omega}')^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') + \frac{\mu}{2} \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 \quad \forall \boldsymbol{\omega}, \boldsymbol{\omega}' \in \Omega\,. \tag{1.5}$$

Definition 6. *A function $(\boldsymbol{\theta}, \boldsymbol{\varphi}) \mapsto \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi})$ is said convex-concave if $\mathcal{L}(\cdot, \boldsymbol{\varphi})$ is convex for all $\boldsymbol{\varphi} \in \Phi$ and $\mathcal{L}(\boldsymbol{\theta}, \cdot)$ is concave for all $\boldsymbol{\theta} \in \Theta$. An $\mathcal{L}$ is said to be μ-strongly convex concave if $(\boldsymbol{\theta}, \boldsymbol{\varphi}) \mapsto \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) - \frac{\mu}{2}\|\boldsymbol{\theta}\|_2^2 + \frac{\mu}{2}\|\boldsymbol{\varphi}\|_2^2$ is convex concave.*

Definition 7. *For $\mu > 0$, an operator $F : \Omega \to \mathbb{R}^d$ is said to be μ-strongly monotone if*

$$(F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}') \geq \mu \|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2\,. \tag{1.6}$$

2 Gradient methods on unconstrained bilinear games

In this section we will prove the results provided in §3, namely Proposition 1, Proposition 2 and Theorem 1. For Proposition 1 and 2 let us recall the context. We wanted to derive properties of some gradient methods on the following simple illustrative example

$$\min_{\theta \in \mathbb{R}} \max_{\phi \in \mathbb{R}} \quad \theta \cdot \phi \tag{2.1}$$

2.1 Proof of Proposition 1

Let us first recall the proposition:

Proposition 1. *The* simultaneous *iterates diverge geometrically and the* alternated *iterates defined in* (3.5) *are bounded but do not converge to 0 as*

$$\text{Simultaneous: } \theta_{t+1}^2+\phi_{t+1}^2 = (1+\eta^2)(\theta_t^2+\phi_t^2)\,, \qquad \text{Alternated: } \theta_t^2+\phi_t^2 = \Theta(\theta_0^2+\phi_0^2) \tag{2.2}$$

where $u_t = \Theta(v_t) \Leftrightarrow \exists \alpha, \beta > 0 : \alpha v_t \leq u_t \leq \beta v_t$.

The uniform average $(\bar{\theta}_t, \bar{\phi}_t) := \frac{1}{t}\sum_{s=0}^{t-1}(\theta_s, \phi_s)$ *of the* simultaneous *updates (resp. the* alternated updates*) diverges (resp. converges to 0) as,*

$$\text{Simultaneous: } \bar{\theta}_t^2+\bar{\phi}_t^2 = \Theta\left(\frac{\theta_0^2+\phi_0^2}{\eta^2 t^2}(1+\eta^2)^t\right), \quad \text{Alternated: } \bar{\theta}_t^2+\bar{\phi}_t^2 = \Theta\left(\frac{\theta_0^2+\phi_0^2}{\eta^2 t^2}\right). \tag{2.3}$$

Proof. Let us start with the *simultaneous* update rule:

$$\begin{cases} \theta_{t+1} = \theta_t - \eta\phi_t \\ \phi_{t+1} = \phi_t + \eta\theta_t\,. \end{cases} \tag{2.4}$$

Then we have,

$$\theta_{t+1}^2 + \phi_{t+1}^2 = (\theta_t - \eta\phi_t)^2 + (\phi_t + \eta\theta_t)^2 \tag{2.5}$$
$$= (1+\eta^2)(\theta_t^2 + \phi_t^2)\,. \tag{2.6}$$

The update rule (2.4) also gives us,

$$\begin{cases} \eta\phi_t = \theta_t - \theta_{t+1} \\ \eta\theta_{t+1} = \phi_{t+1} - \phi_t\,. \end{cases} \tag{2.7}$$

Summing these equation for $0 \leq t \leq T-1$ we get,

$$\bar{\phi}_T^2 + \bar{\theta}_T^2 = (\theta_0 - \theta_T)^2 + (\phi_0 - \phi_T)^2 \tag{2.8}$$
$$= ((1+\eta^2)^T + 1)(\theta_0^2 + \phi_0^2) - 2\theta_0\theta_T - 2\phi_0\phi_T \tag{2.9}$$
$$= \Theta\left((1+\eta^2)^T((\theta_0^2+\phi_0^2)\right) \tag{2.10}$$

Let continue start with the *alternated* update rule

$$\begin{cases} \theta_{t+1} = \theta_t - \eta\phi_t \\ \phi_{t+1} = \phi_t + \eta\theta_{t+1} = \phi_t + \eta(\theta_t - \eta\phi_t) \end{cases} \tag{2.11}$$

Then we have

$$\begin{bmatrix} \theta_{t+1} \\ \phi_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & -\eta \\ \eta & 1-\eta^2 \end{bmatrix} \begin{bmatrix} \theta_t \\ \phi_t \end{bmatrix} \tag{2.12}$$

by simple linear algebra, for $\eta < 2$, the matrix $M := \begin{bmatrix} 1 & -\eta \\ \eta & 1-\eta^2 \end{bmatrix}$ has two complex conjugate eigenvalues which are

$$\lambda_\pm = 1 - \eta \frac{\eta \pm i\sqrt{4-\eta^2}}{2} \tag{2.13}$$

and their squared magnitude is equal to $\det(M) = 1 - \eta^2 + \eta^2 = 1$. We can diagonalize M meaning that there exist P an invertible matrix such that $M = P^{-1}\operatorname{diag}(\lambda_+, \lambda_-)P$. Then, we have

$$\begin{bmatrix} \theta_t \\ \phi_t \end{bmatrix} = M^t \begin{bmatrix} \theta_0 \\ \phi_0 \end{bmatrix} = P^{-1}\operatorname{diag}(\lambda_+^t, \lambda_-^t)P \begin{bmatrix} \theta_0 \\ \phi_0 \end{bmatrix} \tag{2.14}$$

and consequently,

$$\theta_t^2 + \phi_t^2 = \left\| \begin{bmatrix} \theta_t \\ \phi_t \end{bmatrix} \right\|_{\mathbb{C}}^2 = \left\| P^{-1}\operatorname{diag}(\lambda_+^t, \lambda_-^t)P \begin{bmatrix} \theta_0 \\ \phi_0 \end{bmatrix} \right\|_{\mathbb{C}}^2 \leq \|P^{-1}\|\|P\|(\theta_0^2 + \phi_0^2) \tag{2.15}$$

where $\|\cdot\|_{\mathbb{C}}$ is the norm in $\mathbb{C}^2$ and $\|P\| := \max_{u \in \mathbb{C}^2} \frac{\|Pu\|_{\mathbb{C}}}{\|u\|_{\mathbb{C}}}$ is the induced matrix norm. The same way we have

$$\theta_0^2 + \phi_0^2 = \left\| M^{-t} \begin{bmatrix} \theta_t \\ \phi_t \end{bmatrix} \right\|_{\mathbb{C}}^2 = \left\| P^{-1}\operatorname{diag}(\lambda_+^{-t}, \lambda_-^{-t})P \begin{bmatrix} \theta_t \\ \phi_t \end{bmatrix} \right\|_{\mathbb{C}}^2 \leq \|P^{-1}\|\|P\|(\theta_t^2 + \phi_t^2) \tag{2.16}$$

Hence, if $\theta_0^2 + \phi_0^2 > 0$ the sequence (θ_t, ϕ_t) is bounded but do not converge to 0. Moreover the update rule gives us,

$$\begin{cases} \eta\phi_t = \theta_t - \theta_{t+1} \\ \eta\theta_t = \phi_t - \phi_{t-1} \end{cases} \Rightarrow \begin{cases} \dfrac{\eta}{T}\displaystyle\sum_{t=0}^{T-1} \phi_t = \dfrac{\theta_0 - \theta_T}{T} \\ \dfrac{\eta}{T}\displaystyle\sum_{t=0}^{T-1} \theta_t = \dfrac{\phi_{T-1} - \phi_0 + \eta\theta_0}{T} \end{cases} \Rightarrow \begin{cases} \bar{\phi}_T = \dfrac{\theta_0 - \theta_T}{\eta T} \\ \bar{\theta}_T = \dfrac{\phi_{T-1} - \phi_0 + \eta\theta_0}{\eta T} \end{cases} \tag{2.17}$$

Consequently, since $\theta_t^2 + \phi_t^2 = \mathcal{O}(\theta_0^2 + \phi_0^2)$,

$$\sqrt{\bar{\theta}_t^2 + \bar{\phi}_t^2} = \mathcal{O}\left(\frac{\sqrt{\theta_0^2 + \phi_0^2}}{\eta t} \right) \tag{2.18}$$

□

2.2 Implicit and extrapolation method

In this section we will prove a slightly more precise proposition than Proposition 2,

Proposition 2. *The squared norm of the iterates $N_t := \theta_t^2 + \phi_t^2$, where the update rule of θ_t and ϕ_t defined in (3.12), decrease geometrically for any $0 < \eta < 1$ as,*

$$\text{Implicit: } N_{t+1} = \frac{N_t}{1+\eta^2}\,, \quad \text{Extrapolation: } N_{t+1} = (1-\eta^2+\eta^4)N_t\,, \quad \forall t \geq 0 \tag{2.19}$$

Proof. Let us recall the update rule for the implicit method

$$\begin{cases} \theta_{t+1} = \theta_t - \eta\phi_{t+1} \\ \phi_{t+1} = \phi_t + \eta\theta_{t+1} \end{cases} \Rightarrow \begin{cases} (1+\eta^2)\theta_{t+1} = \theta_t - \eta\phi_t \\ (1+\eta^2)\phi_{t+1} = \phi_t + \eta\theta_t \end{cases} \tag{2.20}$$

Then,

$$(1+\eta^2)^2(\theta_{t+1}^2 + \phi_{t+1}^2) = (\theta_t - \eta\phi_t)^2 + (\phi_t + \eta\theta_t)^2 \tag{2.21}$$
$$= \theta_t^2 + \phi_t^2 + +\eta^2(\theta_t^2 + \phi_t^2) \tag{2.22}$$

Implying that

$$\theta_{t+1}^2 + \phi_{t+1}^2 = \frac{\theta_t^2 + \phi_t^2}{1+\eta^2} \tag{2.23}$$

For the extrapolation method we have the update rule

$$\begin{cases} \theta_{t+1} = \theta_t - \eta(\phi_t + \eta\theta_t) \\ \phi_{t+1} = \phi_t + \eta(\theta_t - \eta\phi_t) \end{cases} \tag{2.24}$$

Implying that,

$$\theta_{t+1}^2 + \phi_{t+1}^2 = (\theta_t - \eta(\phi_t + \eta\theta_t))^2 + (\phi_t + \eta(\theta_t - \eta\phi_t))^2 \tag{2.25}$$
$$= \theta_t^2 + \phi_t^2 - 2\eta^2(\theta^2 + \phi^2) + \eta^2((\theta_t - \eta\phi_t)^2 + (\phi_t + \eta\theta_t)^2) \tag{2.26}$$
$$= (1-\eta^2+\eta^4)(\theta_t^2 + \phi_t^2) \tag{2.27}$$

□

2.3 Extrapolation from the past

Let us recall what we call *projected extrapolation form the past*, where we used the notation $\boldsymbol{\omega}_t' = \boldsymbol{\omega}_{t+1/2}$ for compactness,

$$\text{Extrapolation from the past: } \boldsymbol{\omega}_t' = P_\Omega[\boldsymbol{\omega}_t - \eta F(\boldsymbol{\omega}_{t-1}')] \tag{2.28}$$
$$\text{Perform update step: } \boldsymbol{\omega}_{t+1} = P_\Omega[\boldsymbol{\omega}_t - \eta F(\boldsymbol{\omega}_t')] \text{ and store: } F(\boldsymbol{\omega}_t') \tag{2.29}$$

where $P_\Omega[\cdot]$ is the projection onto the constraint set Ω. An operator $F : \Omega \to \mathbb{R}^d$ is said to be μ-strongly monotone if

$$(F(\boldsymbol{\omega}) - F(\boldsymbol{\omega}'))^\top(\boldsymbol{\omega} - \boldsymbol{\omega}') \geq \mu\|\boldsymbol{\omega} - \boldsymbol{\omega}'\|_2^2 \,. \tag{2.30}$$

If F is strongly monotone we can prove the following theorem:

Theorem 1. *If F is μ-strongly monotone (see Appendix A §1 for the definition of strong monotonicity) and L-Lipschitz, then the updates* (3.14) *and* (3.15) *with $\eta = \frac{1}{4L}$ provide linearly converging iterates,*

$$\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2 \leq \left(1 - \frac{\mu}{4L}\right)^t \|\boldsymbol{\omega}_0 - \boldsymbol{\omega}^*\|_2^2\,, \quad \forall t \geq 0\,. \tag{2.31}$$

Proof. In order to prove tis theorem we will prove a slightly more general result,

$$\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}^*\|_2^2 + \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 \leq \left(1 - \frac{\mu}{4L}\right)(\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2 + \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_{t-2}\|_2^2)\,. \tag{2.32}$$

implying that

$$\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}^*\|_2^2 \leq \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}^*\|_2^2 + \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 \leq \left(1 - \frac{\mu}{4L}\right)^t \|\boldsymbol{\omega}_0 - \boldsymbol{\omega}^*\|_2^2\,. \tag{2.33}$$

with the convention that $\boldsymbol{\omega}'_0 = \boldsymbol{\omega}'_{-1} = \boldsymbol{\omega}'_{-2}$.

Let us first proof three technical lemmas,

Lemma 3. *If F is μ-strongly monotone we have,*

$$\mu\left(\|\boldsymbol{\omega}_t - \boldsymbol{\omega}^*\|_2^2 - 2\|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2\right) \leq 2F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}^*)\,, \quad \forall\, \boldsymbol{\omega}^* \in \Omega^*\,. \tag{2.34}$$

Proof. By strong monotonicity and optimality of $\boldsymbol{\omega}^*$,

$$2\mu\|\boldsymbol{\omega}'_t - \boldsymbol{\omega}^*\|_2^2 \leq 2F(\boldsymbol{\omega}^*)^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}^*) + 2\mu\|\boldsymbol{\omega}'_t - \boldsymbol{\omega}^*\|_2^2 \leq 2F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}^*) \tag{2.35}$$

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

Lemma 4. *If F is L-Lipschitz, we have for any $\boldsymbol{\omega} \in \Omega$,*

$$2\eta_t F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}) \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 + \eta_t^2 L^2\|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2\,. \tag{2.36}$$

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

$$\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}) - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t\|_2^2 \tag{2.37}$$

and

$$\|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_{t-1})^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 \,. \tag{2.38}$$

Summing (2.37) and (2.38) we get,

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 &\leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}) && (2.39)\\
&\quad - 2\eta_t F(\boldsymbol{\omega}'_{t-1})^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 && (2.40)\\
&= \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}) - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 \\
&\quad - 2\eta_t (F(\boldsymbol{\omega}'_{t-1}) - F(\boldsymbol{\omega}'_t))^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) \,. && (2.41)
\end{aligned}$$

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

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 &\leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}) + \eta_t^2 \|F(\boldsymbol{\omega}'_{t-1}) - F(\boldsymbol{\omega}'_t)\|_2^2 \\
&\quad + \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 && (2.42)\\
&= \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}) + \eta_t^2 \|F(\boldsymbol{\omega}'_{t-1}) - F(\boldsymbol{\omega}'_t)\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 \\
&\leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}) + \eta_t^2 L^2 \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 \,. && (2.43)
\end{aligned}$$

□

Lemma 5. *For all $t \geq 0$, if we set $\boldsymbol{\omega}'_{-2} = \boldsymbol{\omega}'_{-1} = \boldsymbol{\omega}'_0$ we have*

$$\|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 \leq 4\|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_t\|_2^2 + 4\eta_{t-1}^2 L^2 \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_{t-2}\|_2^2 - \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 \,. \tag{2.44}$$

Proof. We start with $\|a + b\|_2^2 \leq 2\|a\|^2 + 2\|b\|^2$.

$$\|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 \leq 2\|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_t\|_2^2 + 2\|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_{t-1}\|_2^2 \,. \tag{2.45}$$

Moreover, since the projection is contractive we have that

$$\begin{aligned}
\|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_{t-1}\|_2^2 &\leq \|\boldsymbol{\omega}_{t-1} - \eta_{t-1} F(\boldsymbol{\omega}'_{t-1}) - \boldsymbol{\omega}_{t-1} - \eta_{t-1} F(\boldsymbol{\omega}'_{t-2})\|_2^2 && (2.46)\\
&= \eta_{t-1}^2 \|F(\boldsymbol{\omega}'_{t-1}) - F(\boldsymbol{\omega}'_{t-2})\|_2^2 && (2.47)\\
&\leq \eta_{t-1}^2 L^2 \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_{t-2}\|_2^2 \,. && (2.48)
\end{aligned}$$

Combining (2.45) and (2.48) we get,

$$\begin{aligned}
\|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 &= 2\|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 - \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 && (2.49)\\
&\leq 4\|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_t\|_2^2 + 4\|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_{t-1}\|_2^2 - \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 && (2.50)\\
&\leq 4\|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_t\|_2^2 + 4\eta_{t-1}^2 L^2 \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_{t-2}\|_2^2 - \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 \,. && (2.51)
\end{aligned}$$

□

Proof of Theorem 1. Let $\boldsymbol{\omega}^* \in \Omega^*$ be an optimal point of (VIP). Combining Lemma 3 and Lemma 4 we get,

$$\eta_t\mu\left(\|\boldsymbol{\omega}_t-\boldsymbol{\omega}^*\|_2^2-2\|\boldsymbol{\omega}'_t-\boldsymbol{\omega}_t\|_2^2\right) \leq \|\boldsymbol{\omega}_t-\boldsymbol{\omega}^*\|_2^2-\|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}^*\|_2^2+\eta_t^2L^2\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2-\|\boldsymbol{\omega}'_t-\boldsymbol{\omega}_t\|_2^2$$

leading to,

$$\|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}^*\|_2^2 \leq (1-\eta_t\mu)\|\boldsymbol{\omega}_t-\boldsymbol{\omega}^*\|_2^2+\eta_t^2L^2\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2-(1-2\eta_t\mu)\|\boldsymbol{\omega}'_t-\boldsymbol{\omega}_t\|_2^2 \tag{2.52}$$

Now using Lemma 5 we get,

$$\begin{aligned}\|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}^*\|_2^2 \leq (1-\eta_t\mu)\|\boldsymbol{\omega}_t-\boldsymbol{\omega}^*\|_2^2+\eta_t^2L^2(4\eta_{t-1}^2L^2\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_{t-2}\|_2^2-\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2)\\ -(1-2\eta_t\mu-4\eta_t^2L^2)\|\boldsymbol{\omega}'_t-\boldsymbol{\omega}_t\|_2^2\end{aligned} \tag{2.53}$$

Now with $\eta_t = \frac{1}{4L} \leq \frac{1}{4\mu}$ we get,

$$\|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}^*\|_2^2 \leq \left(1-\frac{\mu}{4L}\right)\|\boldsymbol{\omega}_t-\boldsymbol{\omega}^*\|_2^2+\frac{1}{16}\left(\frac{1}{4}\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_{t-2}\|_2^2-\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2\right)$$

Hence, using the fact that $\frac{\mu}{4L} \leq \frac{1}{4}$ we get,

$$\|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}^*\|_2^2+\tfrac{1}{16}\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2 \leq \left(1-\frac{\mu}{4L}\right)\left(\|\boldsymbol{\omega}_t-\boldsymbol{\omega}^*\|_2^2+\tfrac{1}{16}\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_{t-2}\|_2^2\right). \tag{2.54}$$

□

3 More on merit functions

In this section, we will present how to handle *unbounded* constaints set Ω with a more refined *merit function* than (4.2) used in the main paper. Let F the continuous operator and Ω the constraints set associated with the VIP

$$\text{find } \boldsymbol{\omega}^* \in \Omega \quad \text{such that} \quad F(\boldsymbol{\omega}^*)^\top(\boldsymbol{\omega}-\boldsymbol{\omega}^*) \geq 0\,, \;\; \forall \boldsymbol{\omega} \in \Omega\,. \tag{VIP}$$

When the operator F is monotone, we have that $F(\boldsymbol{\omega}^*)^\top(\boldsymbol{\omega}-\boldsymbol{\omega}^*) \leq F(\boldsymbol{\omega})^\top(\boldsymbol{\omega}-\boldsymbol{\omega}^*)\,,\ \forall \boldsymbol{\omega}, \boldsymbol{\omega}^*$. Hence, in this case (VIP) implies a stronger formulation sometimes called *Minty variational inequality* [Crespi et al., 2005]:

$$\text{find } \boldsymbol{\omega}^* \in \Omega \quad \text{such that} \quad F(\boldsymbol{\omega})^\top(\boldsymbol{\omega}-\boldsymbol{\omega}^*) \geq 0\,, \;\; \forall \boldsymbol{\omega} \in \Omega\,. \tag{MVI}$$

This formulation is stronger in the sense that if (MVI) holds for some $\boldsymbol{\omega}^* \in \Omega$, then (VIP) holds too. A *merit function* useful for our analysis can be derived from this formulation. Roughly, a merit function is a convergence measure. More formally, a function $g : \Omega \to \mathbb{R}$ is called a *merit function* if g is non-negative such that $g(\boldsymbol{\omega}) = 0 \Leftrightarrow \boldsymbol{\omega} \in \Omega^*$ [Larsson and Patriksson, 1994]. A way to derive a merit function from (MVI) would be to use $g(\boldsymbol{\omega}^*) = \sup_{\boldsymbol{\omega}\in\Omega} F(\boldsymbol{\omega})^\top(\boldsymbol{\omega}^* - \boldsymbol{\omega})$ which is zero if and only if (MVI) holds for $\boldsymbol{\omega}^*$. To deal with unbounded constraint sets (leading to a potentially infinite valued function outside of the optimal set), we use the *restricted merit function* [Nesterov, 2007]:

$$\mathrm{Err}_R(\boldsymbol{\omega}_t) := \max_{\boldsymbol{\omega}\in\Omega, \|\boldsymbol{\omega}-\boldsymbol{\omega}_0\| \le R} F(\boldsymbol{\omega})^\top(\boldsymbol{\omega}_t - \boldsymbol{\omega})\,. \tag{3.1}$$

This function acts as merit function for (VIP) on the interior of the open ball of radius R around $\boldsymbol{\omega}_0$, as shown in Lemma 1 of Nesterov [2007]. That is, let $\Omega_R := \Omega \cap \{\boldsymbol{\omega} : \|\boldsymbol{\omega} - \boldsymbol{\omega}_0\| < R\}$. Then for any point $\hat{\boldsymbol{\omega}} \in \Omega_R$, we have:

$$\mathrm{Err}_R(\hat{\boldsymbol{\omega}}) = 0 \Leftrightarrow \hat{\boldsymbol{\omega}} \in \Omega^* \cap \Omega_R. \tag{3.2}$$

The reference point $\boldsymbol{\omega}_0$ is arbitrary, but in practice it is usually the initialization point of the algorithm. R has to be big enough to ensure that Ω_R contains a solution. Err_R measures how much (MVI) is violated on the restriction Ω_R. Such merit function is standard in the variational inequality literature. A similar one is used in [Nemirovski, 2004, Juditsky et al., 2011]. When F is derived from the gradients (2.5) of a zero-sum game, we can define a more interpretable merit function. One has to be careful though when extending properties from the minimization setting to the saddle point setting (e.g. the merit function used by Yadav et al. [2018] is vacuous for a bilinear game as explained in App 3.2).

In the appendix we adopt a set of assumptions a little more general than the one in the main paper

Assumption 4.
- *F is* monotone *and Ω is convex and closed.*
- *R is set big enough such that $R > \|\boldsymbol{\omega}_0 - \boldsymbol{\omega}^*\|$ and F is a* monotone *operator.*

Contrary to Assumption 3, in Assumption 4 the constraints set in no longer assumed to be bounded. Assumption 4 is more general than Assumption 3 since that if Ω is bounded then with R set to the diameter of Ω, Assumption 4 is true.

3.1 More general merit functions

In the appendix we will note $\mathrm{Err}_R^{(\mathrm{VI})}$ the *restricted merit function* defined in (3.1). Let us recall its definition,

$$\mathrm{Err}_R^{(\mathrm{VI})}(\boldsymbol{\omega}_t) := \max_{\boldsymbol{\omega}\in\Omega, \|\boldsymbol{\omega}-\boldsymbol{\omega}_0\| \le R} F(\boldsymbol{\omega})^\top(\boldsymbol{\omega}_t - \boldsymbol{\omega})\,. \tag{3.3}$$

When the objective is a saddle point problem i.e.,

$$F(\boldsymbol{\theta}, \boldsymbol{\varphi}) = [\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \;\; -\nabla_{\boldsymbol{\varphi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi})]^\top \tag{3.4}$$

and $\mathcal{L}$ is *convex-concave* (see Definition 6 in §1), we can use another merit function than (3.3) on Ω_R that is more interpretable and more directly related to the cost function of the minimax formulation:

$$\mathrm{Err}_R^{(\mathrm{SP})}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t) := \max_{\substack{\boldsymbol{\varphi} \in \Phi, \boldsymbol{\theta} \in \Theta \\ \|(\boldsymbol{\theta}, \boldsymbol{\varphi}) - (\boldsymbol{\theta}_0, \boldsymbol{\varphi}_0)\| \leq R}} \mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}) - \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}_t) \,. \tag{3.5}$$

In particular, if the equilibrium $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*) \in \Omega^* \cap \Omega_R$ and we have that $\mathcal{L}(\cdot, \boldsymbol{\varphi}^*)$ and $-\mathcal{L}(\boldsymbol{\theta}^*, \cdot)$ are μ-strongly convex (see §1), then the merit function for saddle points upper bounds the distance for $(\boldsymbol{\theta}, \boldsymbol{\varphi}) \in \Omega_R$ to the equilibrium as:

$$\mathrm{Err}_R^{(\mathrm{SP})}(\boldsymbol{\theta}, \boldsymbol{\varphi}) \geq \frac{\mu}{2} (\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2^2 + \|\boldsymbol{\varphi} - \boldsymbol{\varphi}^*\|_2^2) \,. \tag{3.6}$$

In the appendix, we provide our convergence results with the merit functions (3.3) and (3.5), depending on the setup:

$$\mathrm{Err}_R(\boldsymbol{\omega}) := \begin{cases} \mathrm{Err}_R^{(\mathrm{SP})}(\boldsymbol{\omega}) & \text{if } F \text{ is a SP operator (2.5)} \\ \mathrm{Err}_R^{(\mathrm{VI})}(\boldsymbol{\omega}) & \text{otherwise.} \end{cases} \tag{3.7}$$

3.2 On the importance of the merit function

In this section, we illustrate the fact that one has to be careful when extending results and properties from the minimization setting to the minimax setting (and consequently to the variational inequality setting). Another candidate as a merit function for saddle point optimization would be to naturally extend the suboptimality $f(\boldsymbol{\omega}) - f(\boldsymbol{\omega}^*)$ used in standard minimization (i.e. find $\boldsymbol{\omega}^*$ the minimizer of f) to the gap $P(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}^*) - \mathcal{L}(\boldsymbol{\theta}^*, \boldsymbol{\varphi})$. In a previous analysis of a modification of the stochastic gradient descent (SGD) method for GANs, Yadav et al. [2018] gave their convergence rate on P that they called the "primal-dual" gap. Unfortunately, if we do not assume that the function $\mathcal{L}$ is strongly convex-concave (a stronger assumption defined in §1 and which fails for bilinear objective e.g.), P may not be a *merit function*. It can be 0 for a non optimal point, see for instance the discussion on the differences between (3.5) and P in [Gidel et al., 2017, Section 3]. In particular, for the simple 2D bilinear example $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \boldsymbol{\theta} \cdot \boldsymbol{\varphi}$, we have that $\boldsymbol{\theta}^* = \boldsymbol{\varphi}^* = 0$ and thus $P(\boldsymbol{\theta}, \boldsymbol{\varphi}) = 0 \;\; \forall \boldsymbol{\theta}, \boldsymbol{\varphi}$.

3.3 Variational inequalities for non-convex cost functions

When the cost functions defined in (3.1) are non-convex, the operator F is no longer monotone. Nevertheless, (VIP) and (MVI) can still be defined, though a solution to (MVI) is less likely to exist. We note that (VIP) is a local condition for F (as only evaluating F at the points $\boldsymbol{\omega}^*$). On the other hand, an appealing property of (MVI) is that it is a global condition. In the context of minimization of a function f for example (where $F = \nabla f$), if $\boldsymbol{\omega}^*$ solves (MVI) then $\boldsymbol{\omega}^*$ is a *global* minimum of f (and not just a stationary point for the solution of (MVI); see Proposition 2.2 from Crespi et al. [2005]).

A less restrictive way to consider variational inequalities in the non-monotone setting is to use a local version of (MVI). If the cost functions are locally convex around the optimal couple $(\boldsymbol{\theta}^*, \boldsymbol{\varphi}^*)$ and if our iterates eventually fall and stay into that neighborhood, then we can consider our restricted merit function $\mathrm{Err}_R(\cdot)$ with a well suited constant R and apply our convergence results for monotone operators.

4 Another way of implementing extrapolation to SGD

We now introduce another way to combine extrapolation and SGD. This extension is very similar to Alg. 5, the only difference is that it re-uses the minibatch sample of the extrapolation step for the update of the current point. The intuition is that it correlates the estimator of the gradient of the extrapolation step and the one of the update step leading to a better correction of the oscillations which are also due to the stochasticity. One emerging issue (for the analysis) of this method is that since $\boldsymbol{\omega}'_t$ depend on ξ_t, the quantity $F(\boldsymbol{\omega}'_t, \xi_t)$ is a biased estimator of $F(\boldsymbol{\omega}'_t)$.

Algorithm 5 Re-used minibatches for stochastic extrapolation (ReExtraSGD)

```
1: Let $\boldsymbol{\omega}_0 \in \Omega$
2: **for** $t = 0 \dots T-1$ **do**
3:     Sample $\xi_t \sim P$
4:     $\boldsymbol{\omega}'_t := P_\Omega[\boldsymbol{\omega}_t - \eta_t F(\boldsymbol{\omega}_t, \xi_t)]$          ▷ Extrapolation step
5:     $\boldsymbol{\omega}_{t+1} := P_\Omega[\boldsymbol{\omega}_t - \eta_t F\big(\boldsymbol{\omega}'_t, \xi_t\big)]$    ▷ Update step wit the **same** sample
6: **end for**
7: Return $\bar{\boldsymbol{\omega}}_T = \sum_{t=0}^{T-1} \eta_t \boldsymbol{\omega}'_t / \sum_{t=0}^{T-1} \eta_t$
```

Theorem 1. *Assume that $\|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_0\| \leq R$, $\forall t \geq 0$ where $(\boldsymbol{\omega}'_t)_{t\geq 0}$ are the iterates of Alg. 5. Under Assumption 1 and 4, for any $T \geq 1$, Alg. 5 with constant step-size $\eta \leq \frac{1}{\sqrt{2}L}$ has the following convergence properties:*

$$\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{\eta T} + \eta\frac{\sigma^2 + 4L^2(4R^2 + \sigma^2)}{2} \quad \textit{where} \quad \bar{\boldsymbol{\omega}}_T := \frac{1}{T}\sum_{t=0}^{T-1} \boldsymbol{\omega}'_t \,.$$

Particularly, $\eta_t = \frac{\eta}{\sqrt{T}}$ gives $\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{O(1)}{\sqrt{T}}$.

The assumption that the sequence of the iterates provided by the algorithm is bounded is strong, but has been also done for instance in [Yadav et al., 2018]. The proof of this result is provided in §6.

5 Variance comparison between AvgSGD and SGD with prediction method

To compare the variance term of AvgSGD in (4.3) with the one of the *SGD with prediction method* [Yadav et al., 2018], we need to have the same convergence certificate. Fortunately, their proof can be adapted to our convergence criterion (using Lemma 6 in §6), revealing an extra $\sigma^2/2$ in the variance term from their paper. The resulting variance can be summarized with our notation as $(M^2(1+L)+\sigma^2)/2$ where the L is the Lipschitz constant of the operator F. Since $M \gg \sigma$, their variance term is then $1+L$ time larger than the one provided by the AvgSGD method.

6 Proof of Theorems

This section is dedicated on the proof of the theorems provided in this paper in a slightly more general form working with the merit function defined in (3.7). First we prove an additional lemma necessary to the proof of our theorems.

Lemma 6. *Let F be a monotone operator and let $(\boldsymbol{\omega}_t), (\boldsymbol{\omega}'_t), (\boldsymbol{z}_t), (\Delta_t), (\xi_t)$ and (ζ_t) be six random sequences such that, for all $t \geq 0$*

$$2\eta_t F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}'_t - \boldsymbol{\omega}) \leq N_t - N_{t+1} + \eta_t^2(M_1(\boldsymbol{\omega}_t, \xi_t) + M_2(\boldsymbol{\omega}'_t, \zeta_t)) + 2\eta_t\Delta_t^\top(\boldsymbol{z}_t - \boldsymbol{\omega})\,,$$

where $N_t = N(\boldsymbol{\omega}_t, \boldsymbol{\omega}'_{t-1}, \boldsymbol{\omega}'_{t-2}) \geq 0$ and we extend $(\boldsymbol{\omega}'_t)$ with $\boldsymbol{\omega}'_{-2} = \boldsymbol{\omega}'_{-1} = \boldsymbol{\omega}'_0$. Let also assume that with $N_0 \leq R$, $\mathbb{E}[\|\Delta_t\|_2^2] \leq \sigma^2$, $\mathbb{E}[\Delta_t | \boldsymbol{z}_t, \Delta_0, \ldots, \Delta_{t-1}] = 0$, $\mathbb{E}[M_1(\boldsymbol{\omega}_t, \xi_t)] \leq M_1$ and $\mathbb{E}[M_2(\boldsymbol{\omega}'_t, \zeta_t)] \leq M_2$, then,

$$\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{S_T} + \frac{M_1 + M_2 + \sigma^2}{2 S_T} \sum_{t=0}^{T-1} \eta_t^2 \tag{6.1}$$

where $\bar{\boldsymbol{\omega}}_T := \sum_{t=0}^{T-1} \eta_t \boldsymbol{\omega}'_t / S_T$ and $S_T := \sum_{t=0}^{T-1} \eta_t$.

Proof of Lemma 6. We sum (6) for $0 \leq t \leq T-1$ to get,

$$2 \sum_{t=0}^{T-1} \eta_t F(\boldsymbol{\omega}'_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) \leq \sum_{t=0}^{T-1} \Big[(N_t - N_{t+1}) + \eta_t^2 ((M_1(\boldsymbol{\omega}_t, \xi_t) + M_2(\boldsymbol{\omega}'_t, \zeta_t)) + 2\eta_t \Delta_t^\top (\boldsymbol{z}_t - \boldsymbol{\omega}) \Big] . \tag{6.2}$$

We will then upper bound each sum in the right-hand side,

$$\Delta_t^\top (\boldsymbol{z}_t - \boldsymbol{\omega}) = \Delta_t^\top (\boldsymbol{z}_t - \boldsymbol{u}_t) + \Delta_t^\top (\boldsymbol{u}_t - \boldsymbol{\omega})$$

where $\boldsymbol{u}_{t+1} := P_\Omega(\boldsymbol{u}_t - \eta_t \Delta_t)$ and $\boldsymbol{u}_0 := \boldsymbol{\omega}_0$. Then,

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

leading to

$$2\eta_t \Delta_t^\top (\boldsymbol{z}_t - \boldsymbol{\omega}) \leq 2\eta_t \Delta_t^\top (\boldsymbol{z}_t - \boldsymbol{u}_t) + \|\boldsymbol{u}_t - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{u}_{t+1} - \boldsymbol{\omega}\|_2^2 + \eta_t^2 \|\Delta_t\|_2^2 \tag{6.3}$$

Then noticing that $\boldsymbol{z}_0 := \boldsymbol{\omega}_0$, back to (6.2) we get a telescoping sum,

$$2 \sum_{t=0}^{T-1} \eta_t F(\boldsymbol{\omega}'_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) \leq 2N_0 + \sum_{t=0}^{T-1} \Big[\eta_t^2 ((M_1(\boldsymbol{\omega}_t, \xi_t) + M_2(\boldsymbol{\omega}'_t, \zeta_t)) + \|\Delta_t\|_2^2) + 2\eta_t \Delta_t^\top (\boldsymbol{z}_t - \boldsymbol{u}_t) \Big] . \tag{6.4}$$

If F is the operator of a convex-concave saddle point (2.5), we get, with $\boldsymbol{\omega}'_t = (\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)$

$$\begin{aligned} F(\boldsymbol{\omega}'_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) &\geq \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)^\top (\boldsymbol{\theta}_t - \boldsymbol{\theta}) - \nabla_{\boldsymbol{\varphi}} \mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t)^\top (\boldsymbol{\varphi}_t - \boldsymbol{\varphi}) \\ &\geq \mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}) - \mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t) + \mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}_t) - \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}_t) \\ &\qquad \text{(by convexity and concavity)} \\ &= \mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}) - \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}_t) \end{aligned}$$

then by convexity of $\mathcal{L}(\cdot, \boldsymbol{\varphi})$ and concavity of $\mathcal{L}(\boldsymbol{\theta}, \cdot)$, we have that,

$$2S_T \sum_{t=0}^{T-1} \frac{\eta_t}{S_T} F(\boldsymbol{\omega}'_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) \geq 2S_T \sum_{t=0}^{T-1} \frac{\eta_t}{S_T} (\mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{\varphi}) - \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}_t)) \geq \bar{2} S_T (\mathcal{L}(\bar{\boldsymbol{\theta}}_t, \boldsymbol{\varphi}) - \mathcal{L}(\bar{\boldsymbol{\theta}}, \boldsymbol{\varphi}_t)) \tag{6.5}$$

Otherwise if the operator F is just monotone since $F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}'_t-\boldsymbol{\omega}) \geq F(\boldsymbol{\omega}')^\top(\boldsymbol{\omega}'_t-\boldsymbol{\omega})$ we have that

$$2S_T \sum_{t=0}^{T-1} \eta_t F(\boldsymbol{\omega}'_t)^\top(\boldsymbol{\omega}'_t-\boldsymbol{\omega}) \geq 2S_T \sum_{t=0}^{T-1} \eta_t F(\boldsymbol{\omega}')^\top(\boldsymbol{\omega}'_t-\boldsymbol{\omega}) = 2S_T F(\boldsymbol{\omega}')^\top(\bar{\boldsymbol{\omega}}_t-\boldsymbol{\omega}) \quad (6.6)$$

In both cases, we can now maximize the left hand side respect to $\boldsymbol{\omega}$ (since the RHS does not depend on $\boldsymbol{\omega}$) to get,

$$2S_T \operatorname{Err}_R(\bar{\boldsymbol{\omega}}_t) \leq 2R^2 + \sum_{t=0}^{T-1} \left[\eta_t^2((M_1(\boldsymbol{\omega}_t,\xi_t) + M_2(\boldsymbol{\omega}'_t,\zeta_t)) + \|\Delta_t\|_2^2) + 2\eta_t \Delta_t^\top(\boldsymbol{z}_t - \boldsymbol{u}_t)\right]. \quad (6.7)$$

Then taking the expectation, since $\mathbb{E}[\Delta_t|\boldsymbol{z}_t,\boldsymbol{u}_t] = \mathbb{E}[\Delta_t|\boldsymbol{z}_t,\Delta_0,\ldots,\Delta_{t-1}] = 0$, $\mathbb{E}_{\zeta_t}[\|\Delta_t\|_2^2] \leq \sigma^2$, $\mathbb{E}_{\xi_t}[M_1(\boldsymbol{\omega}_t,\xi_t)] \leq M_1$ and $\mathbb{E}_{\zeta_t}[M_2(\boldsymbol{\omega}'_t,\zeta_t)] \leq M_2$, we get that,

$$\mathbb{E}[\operatorname{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{S_T} + \frac{M_1 + M_2 + \sigma^2}{2S_T} \sum_{t=0}^{T-1} \eta_t^2 \quad (6.8)$$

□

6.1 Proof of Thm.2

First let us state Theorem 2 in its general form,

Theorem 2. *Under Assumption 1, 2 and 4, Alg. 1 with constant step-size η has the following convergence rate for all $T \geq 1$,*

$$\mathbb{E}[\operatorname{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{2\eta T} + \eta\frac{M^2+\sigma^2}{2} \quad \textit{where} \quad \bar{\boldsymbol{\omega}}_T := \frac{1}{T}\sum_{t=0}^{T-1} \boldsymbol{\omega}_t\,. \quad (6.9)$$

Particularly, $\eta = \frac{R}{\sqrt{T(M^2+\sigma^2)}}$ *gives* $\mathbb{E}[\operatorname{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R\sqrt{M^2+\sigma^2}}{\sqrt{T}}$.

Proof of Theorem 2. Let any $\boldsymbol{\omega} \in \Omega$ such that $\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}\|_2 \leq R$,

$$\begin{aligned}\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 &= \|P_\Omega(\boldsymbol{\omega}_t - \eta_t F(\boldsymbol{\omega}_t,\xi_t)) - \boldsymbol{\omega}\|_2^2 \\ &\leq \|\boldsymbol{\omega}_t - \eta_t F(\boldsymbol{\omega}_t,\xi_t)) - \boldsymbol{\omega}\|_2^2 \\ &\quad \text{(projections are non-contractive, Lemma 1)} \\ &= \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}_t,\xi_t)^\top(\boldsymbol{\omega}_t - \boldsymbol{\omega}) + \|\eta_t F(\boldsymbol{\omega}_t,\xi_t)\|_2^2\end{aligned}$$

Then we can make appear the quantity $F(\boldsymbol{\omega}_t)^\top(\boldsymbol{\omega}_t - \boldsymbol{\omega})$ on the left-hand side,

$$2\eta_t F(\boldsymbol{\omega}_t)^\top(\boldsymbol{\omega}_t-\boldsymbol{\omega}) \leq \|\boldsymbol{\omega}_t-\boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}\|_2^2 + \eta_t^2\|F(\boldsymbol{\omega}_t,\xi_t)\|_2^2 + 2\eta_t(F(\boldsymbol{\omega}_t) - F(\boldsymbol{\omega}_t,\xi_t))^\top(\boldsymbol{\omega}_t-\boldsymbol{\omega}) \quad (6.10)$$

we can sum (6.10) for $0 \le t \le T-1$ to get,

$$2 \sum_{t=0}^{T-1} \eta_t F(\boldsymbol{\omega}_t)^\top (\boldsymbol{\omega}_t - \boldsymbol{\omega}) \le \sum_{t=0}^{T-1} \Big[(\|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|^2) + \eta_t^2 \|F(\boldsymbol{\omega}_t, \xi_t)\|_2^2 + 2\eta_t \Delta_t^\top (\boldsymbol{\omega}_t - \boldsymbol{\omega}) \Big] \quad (6.11)$$

where we noted $\Delta_t := F(\boldsymbol{\omega}_t) - F(\boldsymbol{\omega}_t, \xi_t)$.
By monotonicity, $F(\boldsymbol{\omega}_t)^\top (\boldsymbol{\omega}_t - \boldsymbol{\omega}) \ge F(\boldsymbol{\omega})^\top (\boldsymbol{\omega}_t - \boldsymbol{\omega})$ we get,

$$2 S_T F(\boldsymbol{\omega})^\top (\bar{\boldsymbol{\omega}}_T - \boldsymbol{\omega}) \le \sum_{t=0}^{T-1} \Big[(\|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|^2) + \eta_t^2 \|F(\boldsymbol{\omega}_t, \xi_t)\|_2^2 + 2\eta_t \Delta_t^\top (\boldsymbol{\omega}_t - \boldsymbol{\omega}) \Big] , \quad (6.12)$$

where $S_T := \sum_{t=0}^{T-1} \eta_t$ and $\bar{\boldsymbol{\omega}}_T := \frac{1}{S_T} \sum_{t=0}^{T-1} \eta_t \boldsymbol{\omega}_t$.
We will then upper bound each sum in the right hand side,

$$\Delta_t^\top (\boldsymbol{\omega}_t - \boldsymbol{\omega}) = \Delta_t^\top (\boldsymbol{\omega}_t - \boldsymbol{u}_t) + \Delta_t^\top (\boldsymbol{u}_t - \boldsymbol{\omega})$$

where $\boldsymbol{u}_{t+1} := P_\Omega(\boldsymbol{u}_t - \eta_t \Delta_t)$ and $\boldsymbol{u}_0 = \boldsymbol{\omega}_0$. Then,

$$\|\boldsymbol{u}_{t+1} - \boldsymbol{\omega}\|_2^2 \le \|\boldsymbol{u}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t \Delta_t^\top (\boldsymbol{u}_t - \boldsymbol{\omega}) + \eta_t^2 \|\Delta_t\|_2^2$$

leading to

$$2\eta_t \Delta_t^\top (\boldsymbol{\omega}_t - \boldsymbol{\omega}) \le 2\eta_t \Delta_t^\top (\boldsymbol{\omega}_t - \boldsymbol{u}_t) + \|\boldsymbol{u}_t - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{u}_{t+1} - \boldsymbol{\omega}\|_2^2 + \eta_t^2 \|\Delta_t\|_2^2 \quad (6.13)$$

Then noticing that $\boldsymbol{u}_0 := \boldsymbol{\omega}_0$, back to (6.12) we get a telescoping sum,

$$\begin{aligned} 2 S_T F(\boldsymbol{\omega})^\top (\bar{\boldsymbol{\omega}}_T - \boldsymbol{\omega}) &\le 2\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}\|^2 + \sum_{t=0}^{T-1} \eta_t^2 (\|F(\boldsymbol{\omega}_t, \xi_t)\|_2^2 + \|\Delta_t\|_2^2) + 2 \sum_{t=0}^{T-1} \eta_t \Delta_t^\top (\boldsymbol{\omega}_t - \boldsymbol{u}_t) \\ &\le 2R + \sum_{t=0}^{T-1} \eta_t^2 (\|F(\boldsymbol{\omega}_t, \xi_t)\|_2^2 + \|\Delta_t\|_2^2) + 2 \sum_{t=0}^{T-1} \eta_t \Delta_t^\top (\boldsymbol{\omega}_t - \boldsymbol{u}_t) \end{aligned}$$

Then the right hand side does not depends on $\boldsymbol{\omega}$, we can maximize over $\boldsymbol{\omega}$ to get,

$$2 S_T \operatorname{Err}_R(\bar{\boldsymbol{\omega}}_T) \le 2R + \sum_{t=0}^{T-1} \eta_t^2 (\|F(\boldsymbol{\omega}_t, \xi_t)\|_2^2 + \|\Delta_t\|_2^2) + 2 \sum_{t=0}^{T-1} \eta_t \Delta_t^\top (\boldsymbol{\omega}_t - \boldsymbol{u}_t) \quad (6.14)$$

Noticing that $\mathbb{E}[\Delta_t | \boldsymbol{\omega}_t, \boldsymbol{u}_t] = 0$ (the estimates of F are unbiased), by Assumption 2 $\mathbb{E}[(\|F(\boldsymbol{\omega}_t, \xi_t)\|_2^2] \le M^2$ and by Assumption 1 $\mathbb{E}[\|\Delta_t\|_2^2] \le \sigma^2$ we get,

$$\mathbb{E}[\operatorname{Err}_R(\bar{\boldsymbol{\omega}}_T)] \le \frac{R}{S_T} + \frac{M^2 + \sigma^2}{2 S_T} \sum_{t=0}^{T-1} \eta_t^2 \quad (6.15)$$

particularly for $\eta_t = \eta$ and $\eta_t = \frac{\eta}{\sqrt{t+1}}$ we respectively get,

$$\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{2R}{\eta T} + \frac{\eta}{2}(M^2 + \sigma^2) \tag{6.16}$$

and

$$\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{4R}{\eta\sqrt{T+1} - 1} + 2\eta \ln(T+1) \frac{M^2 + \sigma^2}{\sqrt{T+1} - 1} \tag{6.17}$$

□

6.2 Proof of Thm.3

Theorem 3. *Under Assumption 1 and 4, if $\mathbb{E}_\xi[F]$ is L-Lipschitz, then Alg. 2 with a constant step-size $\eta \leq \frac{1}{\sqrt{3}L}$ has the following convergence rate for any $T \geq 1$,*

$$\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{\eta T} + \frac{7}{2}\eta\sigma^2 \quad \textit{where} \quad \bar{\boldsymbol{\omega}}_T := \frac{1}{T}\sum_{t=0}^{T-1} \boldsymbol{\omega}'_t \,. \tag{6.18}$$

Particularly, $\eta = \frac{\sqrt{2}R}{\sigma\sqrt{7T}}$ gives $\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{\sqrt{14}R\sigma}{\sqrt{T}}$.

Proof of Thm.3. Let any $\boldsymbol{\omega} \in \Omega$ such that $\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}\|_2 \leq R$. Then, the update rules become $\boldsymbol{\omega}_{t+1} = P_\Omega(\boldsymbol{\omega}_t - \eta_t F(\boldsymbol{\omega}'_t, \zeta_t))$ and $\boldsymbol{\omega}'_t = P_\Omega(\boldsymbol{\omega}_t - \eta F(\boldsymbol{\omega}_t, \xi_t))$. We start by applying Lemma 2 for $(\boldsymbol{\omega}, \boldsymbol{u}, \boldsymbol{\omega}', \boldsymbol{\omega}^+) = (\boldsymbol{\omega}_t, -\eta F(\boldsymbol{\omega}'_t, \zeta_t), \boldsymbol{\omega}, \boldsymbol{\omega}_{t+1})$ and $(\boldsymbol{\omega}, \boldsymbol{u}, \boldsymbol{\omega}', \boldsymbol{\omega}^+) = (\boldsymbol{\omega}_t, -\eta_t F(\boldsymbol{\omega}_t, \xi_t), \boldsymbol{\omega}_{t+1}, \boldsymbol{\omega}'_t)$,

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 &\leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t, \zeta_t)^\top (\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}) - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t\|_2^2 \\
\|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 &\leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}_t, \xi_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2
\end{aligned}$$

Then, summing them we get

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 &- 2\eta_t F(\boldsymbol{\omega}'_t, \zeta_t)^\top (\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}) \\
&- 2\eta_t F(\boldsymbol{\omega}_t, \xi_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_t\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}'_t\|_2^2
\end{aligned} \tag{6.19}$$

leading to

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 &- 2\eta_t F(\boldsymbol{\omega}'_t, \zeta_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) \\
&+ 2\eta_t (F(\boldsymbol{\omega}'_t, \zeta_t) - F(\boldsymbol{\omega}_t, \xi_t))^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_t\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}'_t\|_2^2
\end{aligned}$$

Then with $2\boldsymbol{a}^\top \boldsymbol{b} \leq \|\boldsymbol{a}\|_2^2 + \|\boldsymbol{b}\|_2^2$ we get

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 &- 2\eta_t F(\boldsymbol{\omega}'_t, \zeta_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) \\
&- \|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_t\|_2^2 + \eta_t^2 \|F(\boldsymbol{\omega}'_t, \zeta_t) - F(\boldsymbol{\omega}_t, \xi_t)\|_2^2
\end{aligned}$$

Using the inequality $\|\boldsymbol{a}+\boldsymbol{b}+\boldsymbol{c}\|_2^2 \leq 3(\|\boldsymbol{a}\|_2^2+\|\boldsymbol{b}\|_2^2+\|\boldsymbol{c}\|_2^2)$ we get,

$$\begin{aligned}\|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t-\boldsymbol{\omega}\|_2^2 &- 2\eta_t F(\boldsymbol{\omega}_t',\zeta_t)^\top(\boldsymbol{\omega}_t'-\boldsymbol{\omega}) - \|\boldsymbol{\omega}_t-\boldsymbol{\omega}_t'\|_2^2 \\ &+ 3\eta_t^2(\|F(\boldsymbol{\omega}_t)-F(\boldsymbol{\omega}_t,\xi_t)\|_2^2 + \|F(\boldsymbol{\omega}_t')-F(\boldsymbol{\omega}_t',\zeta_t)\|_2^2 + \|F(\boldsymbol{\omega}_t')-F(\boldsymbol{\omega}_t)\|_2^2)\end{aligned}$$

Then we can use the L-Lipschitzness of F to get,

$$\begin{aligned}\|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t-\boldsymbol{\omega}\|_2^2 &- 2\eta_t F(\boldsymbol{\omega}_t',\zeta_t)^\top(\boldsymbol{\omega}_t'-\boldsymbol{\omega}) - \|\boldsymbol{\omega}_t-\boldsymbol{\omega}_t'\|_2^2 \\ &+ 3\eta_t^2(\|F(\boldsymbol{\omega}_t)-F(\boldsymbol{\omega}_t,\xi_t)\|_2^2 + \|F(\boldsymbol{\omega}_t')-F(\boldsymbol{\omega}_t',\zeta_t)\|_2^2 + L^2\|\boldsymbol{\omega}_t-\boldsymbol{\omega}_t'\|_2^2)\end{aligned}$$

As we restricted the step-size to $\eta_t \leq \frac{1}{\sqrt{3}L}$ we get,

$$\begin{aligned}2\eta_t F(\boldsymbol{\omega}_t')^\top(\boldsymbol{\omega}_t'-\boldsymbol{\omega}) \leq \|\boldsymbol{\omega}_t-\boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}\|_2^2 &+ 2\eta_t(F(\boldsymbol{\omega}_t')-F(\boldsymbol{\omega}_t',\zeta_t))^\top(\boldsymbol{\omega}_t'-\boldsymbol{\omega}) \\ &+ 3\eta_t^2\|F(\boldsymbol{\omega}_t)-F(\boldsymbol{\omega}_t,\xi_t)\|_2^2 + 3\eta_t^2\|F(\boldsymbol{\omega}_t')-F(\boldsymbol{\omega}_t',\zeta_t)\|_2^2\end{aligned}$$

We get a particular case of (6) so we can use Lemma 6 where $N_t = \|\boldsymbol{\omega}_t-\boldsymbol{\omega}\|_2^2$, $M_1(\boldsymbol{\omega}_t,\xi_t) = 3\|F(\boldsymbol{\omega}_t)-F(\boldsymbol{\omega}_t,\xi_t)\|_2^2$, $M_2(\boldsymbol{\omega}_t',\zeta_t) = 3\|F(\boldsymbol{\omega}_t')-F(\boldsymbol{\omega}_t',\zeta_t)\|_2^2$, $\Delta_t = F(\boldsymbol{\omega}_t')-F(\boldsymbol{\omega}_t',\zeta_t)$ and $\boldsymbol{z}_t = \boldsymbol{\omega}_t'$. By Assumption 1, $M_1 = M_2 = 3\sigma^2$ and by the fact that $\mathbb{E}[F(\boldsymbol{\omega}_t')-F(\boldsymbol{\omega}_t',\zeta_t)\,|\boldsymbol{\omega}_t',\Delta_0,\ldots,\Delta_{t-1}] = \mathbb{E}[\mathbb{E}[F(\boldsymbol{\omega}_t')-F(\boldsymbol{\omega}_t',\zeta_t)\,|\boldsymbol{\omega}_t']|\Delta_0,\ldots,\Delta_{t-1}] = 0$ the hypothesis of Lemma 6 hold and we get,

$$\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{S_T} + \frac{7\sigma^2}{2S_T}\sum_{t=0}^{T-1}\eta_t^2 \tag{6.20}$$

□

6.3 Proof of Thm. 4

Theorem 4. *Under Assumption 1, if $\mathbb{E}_\xi[F]$ is L-Lipschitz, then AvgPastExtraSGD (Alg. 3) with a constant step-size $\eta \leq \frac{1}{2\sqrt{3}L}$ has the following convergence rate for any $T \geq 1$,*

$$\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{\eta T} + \frac{13}{2}\eta\sigma^2 \quad \textit{where} \quad \bar{\boldsymbol{\omega}}_T := \frac{1}{T}\sum_{t=0}^{T-1}\boldsymbol{\omega}_t'\,. \tag{6.21}$$

Particularly, $\eta = \frac{\sqrt{2}R}{\sigma\sqrt{7T}}$ gives $\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{\sqrt{14}R\sigma}{\sqrt{T}}$.

First let us recall the update rule

$$\begin{cases}\boldsymbol{\omega}_{t+1} = P_\Omega[\boldsymbol{\omega}_t - \eta_t F(\boldsymbol{\omega}_t',\xi_t)] \\ \boldsymbol{\omega}_{t+1}' = P_\Omega[\boldsymbol{\omega}_{t+1} - \eta_{t+1}F(\boldsymbol{\omega}_t',\xi_t)]\,.\end{cases} \tag{6.22}$$

Lemma 7. *We have for any $\boldsymbol{\omega} \in \Omega$,*

$$2\eta F(\boldsymbol{\omega}'_t, \xi_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 + 3\eta_t^2 L^2 \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 + 3\eta_t^2 \Big[\|F(\boldsymbol{\omega}'_{t-1}, \xi_{t-1}) - F(\boldsymbol{\omega}'_{t-1})\|_2^2 + \|F(\boldsymbol{\omega}'_t) - F(\boldsymbol{\omega}'_t, \xi_t)\|_2^2\Big] . \tag{6.23}$$

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

$$\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t, \xi_t)^\top (\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}) - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t\|_2^2 \tag{6.24}$$

and

$$\|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_{t-1}, \xi_{t-1})^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 . \tag{6.25}$$

Summing (6.24) and (6.25) we get,

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 &\leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t, \xi_t)^\top (\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}) && (6.26)\\
&\quad - 2\eta_t F(\boldsymbol{\omega}'_{t-1}, \xi_{t-1})^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 && (6.27)\\
&= \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t, \xi_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 && (6.28)\\
&\quad - 2\eta_t (F(\boldsymbol{\omega}'_{t-1}, \xi_{t-1}) - F(\boldsymbol{\omega}'_t, \xi_t))^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) . && (6.29)
\end{aligned}$$

Then, we can use the inequality of arithmetic and geometric means $2a^\top b \leq \|a\|_2^2 + \|b\|_2^2$ to get,

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 &\leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t, \xi_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) + \eta_t^2 \|F(\boldsymbol{\omega}'_{t-1}, \xi_{t-1}) - F(\boldsymbol{\omega}'_t, \xi_t)\|_2^2 \\
&\quad + \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 && (6.30)\\
&= \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t, \xi_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) && (6.31)\\
&\quad + \eta_t^2 \|F(\boldsymbol{\omega}'_{t-1}, \xi_{t-1}) - F(\boldsymbol{\omega}'_t, \xi_t)\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 . && (6.32)
\end{aligned}$$

Using the inequality $\|\boldsymbol{a} + \boldsymbol{b} + \boldsymbol{c}\|_2^2 \leq 3(\|\boldsymbol{a}\|_2^2 + \|\boldsymbol{b}\|_2^2 + \|\boldsymbol{c}\|_2^2)$ we get,

$$\begin{aligned}
\|F(\boldsymbol{\omega}'_{t-1}, \xi_{t-1}) - F(\boldsymbol{\omega}'_t, \xi_t)\|_2^2 &\leq 3(\|F(\boldsymbol{\omega}'_{t-1}, \xi_{t-1}) - F(\boldsymbol{\omega}'_{t-1})\|_2^2 + \|F(\boldsymbol{\omega}'_{t-1}) - F(\boldsymbol{\omega}'_t)\|_2^2 \\
&\quad + \|F(\boldsymbol{\omega}'_t) - F(\boldsymbol{\omega}'_t, \xi_t)\|_2^2) && (6.33)\\
&\leq 3(\|F(\boldsymbol{\omega}'_{t-1}, \xi_{t-1}) - F(\boldsymbol{\omega}'_{t-1})\|_2^2 + L^2 \|\boldsymbol{\omega}'_{t-1} - \boldsymbol{\omega}'_t\|_2^2 \\
&\quad + \|F(\boldsymbol{\omega}'_t) - F(\boldsymbol{\omega}'_t, \xi_t)\|_2^2) , && (6.34)
\end{aligned}$$

where we used the L-Lipschitzness of F for the last inequality.

Combining (6.32) with (6.34) we get,

$$\|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}\|_2^2 \le \|\boldsymbol{\omega}_t-\boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t,\xi_t)^\top(\boldsymbol{\omega}'_t-\boldsymbol{\omega}) - \|\boldsymbol{\omega}'_t-\boldsymbol{\omega}_t\|_2^2 + 3\eta_t^2 L^2\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2 + 3\eta_t^2\Big[\|F(\boldsymbol{\omega}'_{t-1},\xi_{t-1})-F(\boldsymbol{\omega}'_{t-1})\|_2^2+\|F(\boldsymbol{\omega}'_t)-F(\boldsymbol{\omega}'_t,\xi_t)\|_2^2\Big]. \tag{6.35}$$

□

Lemma 8. *For all $t \ge 0$, if we set $\boldsymbol{\omega}'_{-2}=\boldsymbol{\omega}'_{-1}=\boldsymbol{\omega}'_0$ we have*

$$\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2 \le 4\|\boldsymbol{\omega}_t-\boldsymbol{\omega}'_t\|_2^2 + 12\eta_{t-1}^2\Big(\|F(\boldsymbol{\omega}'_{t-1},\xi_{t-1})-F(\boldsymbol{\omega}'_{t-1})\|_2^2 + L^2\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_{t-2}\|_2^2 + \|F(\boldsymbol{\omega}'_{t-2})-F(\boldsymbol{\omega}'_{t-2},\xi_{t-2})\|_2^2\Big) - \|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2. \tag{6.36}$$

Proof. We start with $\|a+b\|_2^2 \le 2\|a\|^2+2\|b\|^2$.

$$\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2 \le 2\|\boldsymbol{\omega}_t-\boldsymbol{\omega}'_t\|_2^2 + 2\|\boldsymbol{\omega}_t-\boldsymbol{\omega}'_{t-1}\|_2^2. \tag{6.37}$$

Moreover, since the projection is contractive we have that

$$\|\boldsymbol{\omega}_t-\boldsymbol{\omega}'_{t-1}\|_2^2 \le \|\boldsymbol{\omega}_{t-1}-\eta_{t-1}F(\boldsymbol{\omega}'_{t-1},\xi_{t-1})-\boldsymbol{\omega}_{t-1}-\eta_{t-1}F(\boldsymbol{\omega}'_{t-2},\xi_{t-2})\|_2^2 \tag{6.38}$$
$$= \eta_{t-1}^2\|F(\boldsymbol{\omega}'_{t-1},\xi_{t-1})-F(\boldsymbol{\omega}'_{t-2},\xi_{t-2})\|_2^2 \tag{6.39}$$
$$\le 3\eta_{t-1}^2\Big(\|F(\boldsymbol{\omega}'_{t-1},\xi_{t-1})-F(\boldsymbol{\omega}'_{t-1})\|_2^2 + L^2\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_{t-2}\|_2^2 + \|F(\boldsymbol{\omega}'_{t-2})-F(\boldsymbol{\omega}'_{t-2},\xi_{t-2})\|_2^2\Big). \tag{6.40}$$

where in the last line we used the same inequality as in (6.34). Combining (6.36) and (6.40) we get,

$$\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2 = 2\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2 - \|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2 \tag{6.41}$$
$$\le 4\|\boldsymbol{\omega}_t-\boldsymbol{\omega}'_t\|_2^2 + 4\|\boldsymbol{\omega}_t-\boldsymbol{\omega}'_{t-1}\|_2^2 - \|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2 \tag{6.42}$$
$$\le 4\|\boldsymbol{\omega}_t-\boldsymbol{\omega}'_t\|_2^2 + 12\eta_{t-1}^2\Big(\|F(\boldsymbol{\omega}'_{t-1},\xi_{t-1})-F(\boldsymbol{\omega}'_{t-1})\|_2^2 + L^2\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_{t-2}\|_2^2 + \|F(\boldsymbol{\omega}'_{t-2})-F(\boldsymbol{\omega}'_{t-2},\xi_{t-2})\|_2^2\Big) - \|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2. \tag{6.43}$$

□

Proof of Theorem 4. Combining Lemma 8 and Lemma 7 we get,

$$2\eta_t F(\boldsymbol{\omega}'_t,\xi_t)^\top(\boldsymbol{\omega}'_t-\boldsymbol{\omega}) \le \|\boldsymbol{\omega}_t-\boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1}-\boldsymbol{\omega}\|_2^2 + 36\eta_t^2\eta_{t-1}^2L^2\Big(\|F(\boldsymbol{\omega}'_{t-1},\xi_{t-1})-F(\boldsymbol{\omega}'_{t-1})\|_2^2 + L^2\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_{t-2}\|_2^2 + \|F(\boldsymbol{\omega}'_{t-2})-F(\boldsymbol{\omega}'_{t-2},\xi_{t-2})\|_2^2\Big) - 3\eta_t^2L^2\|\boldsymbol{\omega}'_{t-1}-\boldsymbol{\omega}'_t\|_2^2 + (12\eta_t^2L^2-1)\|\boldsymbol{\omega}'_t-\boldsymbol{\omega}_t\|_2^2 + 3\eta_t^2\Big[\|F(\boldsymbol{\omega}'_{t-1},\xi_{t-1})-F(\boldsymbol{\omega}'_{t-1})\|_2^2 + \|F(\boldsymbol{\omega}'_t)-F(\boldsymbol{\omega}'_t,\xi_t)\|_2^2\Big]. \tag{6.44}$$

Then for $\eta_t \leq \frac{1}{2\sqrt{3}L}$ we have $36\eta_t^2\eta_{t-1}^2 L^4 \leq 3\eta_{t-1}^2 L^2$,

$$\begin{aligned}
2\eta_t F(\boldsymbol{\omega}_t')^\top(\boldsymbol{\omega}_t' - \boldsymbol{\omega}) \leq \|&\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \\
&+ 3L^2(\eta_{t-1}^2\|\boldsymbol{\omega}_{t-1}' - \boldsymbol{\omega}_{t-2}'\|_2^2 - \eta_t^2\|\boldsymbol{\omega}_{t-1}' - \boldsymbol{\omega}_t'\|_2^2) \\
&+ 2\eta_t(F(\boldsymbol{\omega}_t') - F(\boldsymbol{\omega}_t', \xi_t))^\top(\boldsymbol{\omega}_t' - \boldsymbol{\omega}) \\
&+ 3\eta_t^2\Big[\|F(\boldsymbol{\omega}_{t-2}', \xi_{t-2}) - F(\boldsymbol{\omega}_{t-2}')\|_2^2 + 2\|F(\boldsymbol{\omega}_{t-1}', \xi_{t-1}) - F(\boldsymbol{\omega}_{t-1}')\|_2^2 \\
&\quad + \|F(\boldsymbol{\omega}_t') - F(\boldsymbol{\omega}_t', \xi_t)\|_2^2\Big]\,. \qquad (6.45)
\end{aligned}$$

We can then use Lemma 6 where

$$\begin{aligned}
N_t &= \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 + 3L^3\eta_{t-1}\|\boldsymbol{\omega}_{t-1}' - \boldsymbol{\omega}_{t-2}'\|_2^2, \\
M_1(\boldsymbol{\omega}_t, \xi_t) &= 0 \\
M_2(\boldsymbol{\omega}_t', \xi_t) &= 3\|F(\boldsymbol{\omega}_t') - F(\boldsymbol{\omega}_t', \xi_t)\|_2^2 + 6\|F(\boldsymbol{\omega}_{t-1}') - F(\boldsymbol{\omega}_{t-1}', \xi_{t-1})\|_2^2 \\
&\quad + 3\|F(\boldsymbol{\omega}_{t-2}') - F(\boldsymbol{\omega}_{t-2}', \xi_{t-2})\|_2^2 \\
\Delta_t &= F(\boldsymbol{\omega}_t') - F(\boldsymbol{\omega}_t', \xi_t) \\
\boldsymbol{z}_t &= \boldsymbol{\omega}_t'\,.
\end{aligned}$$

By Assumption 1, $M_2 = 12\sigma^2$ and by the fact that $\mathbb{E}[F(\boldsymbol{\omega}_t') - F(\boldsymbol{\omega}_t', \xi_t)\,|\,\boldsymbol{\omega}_t', \Delta_0, \ldots, \Delta_{t-1}] = \mathbb{E}[\mathbb{E}[F(\boldsymbol{\omega}$
the hypothesis of Lemma 6 hold and we get,

$$\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{S_T} + \frac{13\sigma^2}{2S_T}\sum_{t=0}^{T-1}\eta_t^2 \qquad (6.46)$$

□

6.4 Proof of Theorem 1

Theorem 1 has been introduced in §4. This theorem is about Algorithm 5 which consists in another way to implement extrapolation to SGD. Let us first restate this theorem,

Theorem 1. *Assume that $\|\boldsymbol{\omega}_t' - \boldsymbol{\omega}_0\| \leq R$, $\forall t \geq 0$ where $(\boldsymbol{\omega}_t')_{t\geq 0}$ are the iterates of Alg. 5. Under Assumption 1 and 4, for any $T \geq 1$, Alg. 5 with constant step-size $\eta \leq \frac{1}{\sqrt{2}L}$ has the following convergence properties:*

$$\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{\eta T} + \eta\frac{\sigma^2 + 4L^2(4R^2 + \sigma^2)}{2} \quad \textit{where} \quad \bar{\boldsymbol{\omega}}_T := \frac{1}{T}\sum_{t=0}^{T-1}\boldsymbol{\omega}_t'\,.$$

Particularly, $\eta_t = \frac{\eta}{\sqrt{T}}$ gives $\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{O(1)}{\sqrt{T}}$.

Proof of Thm. 1. Let any $\boldsymbol{\omega} \in \Omega$ such that $\|\boldsymbol{\omega}_0 - \boldsymbol{\omega}\|_2 \leq R$. Then, the update rules become $\boldsymbol{\omega}_{t+1} = P_\Omega(\boldsymbol{\omega}_t - \eta_t F(\boldsymbol{\omega}'_t, \xi_t))$ and $\boldsymbol{\omega}'_t = P_\Omega(\boldsymbol{\omega}_t - \eta F(\boldsymbol{\omega}_t, \xi_t))$. We start the same way as the proof of Thm. 3 by applying Lemma 2 for $(\boldsymbol{\omega}, \boldsymbol{u}, \boldsymbol{\omega}', \boldsymbol{\omega}^*) = (\boldsymbol{\omega}_t, -\eta F(\boldsymbol{\omega}'_t, \xi_t), \boldsymbol{\omega}, \boldsymbol{\omega}_{t+1})$ and $(\boldsymbol{\omega}, \boldsymbol{u}, \boldsymbol{\omega}', \boldsymbol{\omega}^+) = (\boldsymbol{\omega}_t, -\eta_t F(\boldsymbol{\omega}_t, \xi_t), \boldsymbol{\omega}_{t+1}, \boldsymbol{\omega}'_t)$,

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 &\leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t, \xi_t)^\top (\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}) - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}_t\|_2^2 \\
\|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}\|_2^2 &\leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}_{t+1}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}_t, \xi_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2
\end{aligned}$$

Then, summing them we get

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 &- 2\eta_t F(\boldsymbol{\omega}'_t, \xi_t)^\top (\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}) \\
&- 2\eta_t F(\boldsymbol{\omega}_t, \xi_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}'_t\|_2^2 \quad (6.47)
\end{aligned}$$

leading to

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 &- 2\eta_t F(\boldsymbol{\omega}'_t, \xi_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) \\
&+ 2\eta_t (F(\boldsymbol{\omega}'_t, \xi_t) - F(\boldsymbol{\omega}_t, \xi_t))^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}_{t+1}) - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}'_t\|_2^2
\end{aligned}$$

Then with $2\boldsymbol{a}^\top \boldsymbol{b} \leq \|\boldsymbol{a}\|_2^2 + \|\boldsymbol{b}\|_2^2$ we get

$$\begin{aligned}
\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t, \xi_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) \\
+\eta_t^2 \|F(\boldsymbol{\omega}'_t, \xi_t) - F(\boldsymbol{\omega}_t, \xi_t)\|_2^2 - \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2^2
\end{aligned}$$

Using the Lipschitzness assumption we get

$$\|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - 2\eta_t F(\boldsymbol{\omega}'_t, \xi_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) + (\eta_t^2 L^2 - 1)\|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_t\|_2^2$$

Then we add $2\eta_t F(\boldsymbol{\omega}'_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega})$ in both sides to get,

$$\begin{aligned}
2\eta_t F(\boldsymbol{\omega}'_t)^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) &\leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 \\
&\quad - 2\eta_t (F(\boldsymbol{\omega}'_t, \xi_t) - F(\boldsymbol{\omega}'_t))^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega}) + (\eta_t^2 L^2 - 1)\|\boldsymbol{\omega}_t - \boldsymbol{\omega}'_t\|_2^2 \quad (6.48)
\end{aligned}$$

Here, unfortunately we cannot use Lemma 6 because $F(\boldsymbol{\omega}'_t, \xi_t)$ is biased. We will then deal with the quantity $A = (F(\boldsymbol{\omega}'_t, \xi_t) - F(\boldsymbol{\omega}'))^\top (\boldsymbol{\omega}'_t - \boldsymbol{\omega})$. We have that,

$$\begin{aligned}
A &= (F(\boldsymbol{\omega}'_t, \xi_t) - F(\boldsymbol{\omega}_t, \xi_t))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}'_t) + (F(\boldsymbol{\omega}_t) - F(\boldsymbol{\omega}'_t))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}'_t) \\
&\quad + (F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t))^\top (\boldsymbol{\omega}_t - \boldsymbol{\omega}'_t) + (F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}_t) \\
&\leq 2L\|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2 \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}\|_2 + \|F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t)\| \|\boldsymbol{\omega}'_t - \boldsymbol{\omega}_t\|_2 \\
&\quad + (F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t))^\top (\boldsymbol{\omega} - \boldsymbol{\omega}_t) \\
&\qquad \text{(Using Cauchy-Schwarz and the } L\text{-Lip of } F\text{)}
\end{aligned}$$

Then using $2\|a\|\|b\| \leq \delta\|a\|_2^2 + \frac{1}{\delta}\|b\|_2^2$, for $\delta = 4$,

$$\begin{aligned}-2\eta_t(F(\boldsymbol{\omega}_t', \xi_t) - F(\boldsymbol{\omega}_t'))^\top(\boldsymbol{\omega}_t' - \boldsymbol{\omega}) \leq \frac{1}{2}\|\boldsymbol{\omega}_t' - \boldsymbol{\omega}_t\|^2 &+ 8\eta_t^2 L^2\|\boldsymbol{\omega}_t' - \boldsymbol{\omega}\|_2^2 \\ &+ 4\eta_t^2\|F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t)\|_2^2 \\ &+ \frac{1}{4}\|\boldsymbol{\omega}_t' - \boldsymbol{\omega}_t\|_2^2 + 2\eta_t(F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t))^\top(\boldsymbol{\omega} - \boldsymbol{\omega}_t)\end{aligned}$$

leading to,

$$\begin{aligned}2\eta_t F(\boldsymbol{\omega}_t')^\top(\boldsymbol{\omega}_t' - \boldsymbol{\omega}) \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 &+ 2\eta_t(F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t))^\top(\boldsymbol{\omega} - \boldsymbol{\omega}_t) \\ &+ (\eta_t^2 L^2 - \tfrac{1}{4})\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_t'\|_2^2 \\ &+ 4\eta_t^2(2L^2\|\boldsymbol{\omega}_t' - \boldsymbol{\omega}\|_2^2 + \|F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t)\|_2^2)\end{aligned}$$

If one assumes finally that $\|\boldsymbol{\omega}_t' - \boldsymbol{\omega}_0\|_2 \leq R$ (assumption of the theorem) and that $\eta_t \leq \frac{1}{2L}$ we get,

$$\begin{aligned}2\eta_t F(\boldsymbol{\omega}_t')^\top(\boldsymbol{\omega}_t' - \boldsymbol{\omega}) \leq \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2 - \|\boldsymbol{\omega}_{t+1} - \boldsymbol{\omega}\|_2^2 &+ 2\eta_t(F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t))^\top(\boldsymbol{\omega} - \boldsymbol{\omega}_t) \\ &+ 4\eta_t^2(4L^2R^2 + \|F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t)\|_2^2)\end{aligned}$$

where we used that $\|\boldsymbol{\omega}_t' - \boldsymbol{\omega}\|_2 \leq \|\boldsymbol{\omega}_t' - \boldsymbol{\omega}_0\|_2 + \|\boldsymbol{\omega}_0 - \boldsymbol{\omega}\|_2 \leq 2R$. Once again this equation is a particular case of Lemma 6 where $N_t = \|\boldsymbol{\omega}_t - \boldsymbol{\omega}\|_2^2$, $M_1(\boldsymbol{\omega}_t, \xi_t) = 4(4L^2R^2 + \|F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t)\|_2^2)$, $M_2(\boldsymbol{\omega}_t', \zeta_t) = 0$, $\boldsymbol{z}_t = \boldsymbol{\omega}_t$ and $\Delta_t = F(\boldsymbol{\omega}_t, \xi_t) - F(\boldsymbol{\omega}_t)$. By Assumption 1 $\mathbb{E}[M_1(\boldsymbol{\omega}_t, \xi_t)] \leq 16L^2R^2 + 4\sigma^2$ and $\mathbb{E}[\Delta_t | \boldsymbol{\omega}_t, \Delta_0, \ldots, \Delta_{t-1}] = \mathbb{E}[\mathbb{E}[\Delta_t|\boldsymbol{\omega}_t]|\Delta_0, \ldots, \Delta_{t-1}] = 0$ so we can use Lemma 6 and get,

$$\mathbb{E}[\mathrm{Err}_R(\bar{\boldsymbol{\omega}}_T)] \leq \frac{R^2}{S_T} + \frac{\sigma^2 + 16L^2R^2 + 4\sigma^2}{2S_T}\sum_{t=0}^{T-1}\eta_t^2\,. \tag{6.49}$$

□

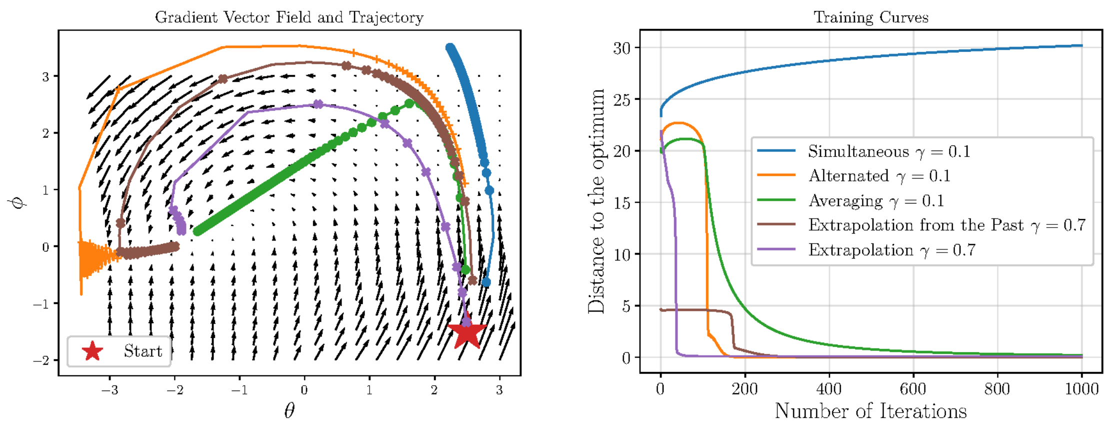


Figure A.1: Comparison of five algorithms (described in Section 3) on the non-convex gan objective (7.1), using the optimal step-size for each method. **Left**: The gradient vector field and the dynamics of the different methods. **Right**:The distance to the optimum as a function of the number of iterations for each methods.

7 Additional experimental results

7.1 Toy non-convex GAN (2D and deterministic)

We now consider a task similar to [Mescheder et al., 2018] where the discriminator is linear $D_{\boldsymbol{\varphi}}(\boldsymbol{\omega}) = \boldsymbol{\varphi}^T\boldsymbol{\omega}$, the generator is a Dirac distribution at $\boldsymbol{\theta}$, $q_{\boldsymbol{\theta}} = \delta_{\boldsymbol{\theta}}$ and the distribution we try to match is also a Dirac at $\boldsymbol{\omega}^*$, $p = \delta_{\boldsymbol{\omega}^*}$. The minimax formulation from Goodfellow et al. [2014] gives:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\varphi}} -\log\left(1 + e^{-\boldsymbol{\varphi}^T\boldsymbol{\omega}^*}\right) - \log\left(1 + e^{\boldsymbol{\varphi}^T\boldsymbol{\theta}}\right) \tag{7.1}$$

Note that as observed by Nagarajan and Kolter [2017], this objective is concave-concave, making it hard to optimize. We compare the methods on this objective where we take $\boldsymbol{\omega}^* = -2$, thus the position of the equilibrium is shifted towards the position $(\boldsymbol{\theta}, \boldsymbol{\varphi}) = (-2, 0)$. The convergence and the gradient vector field are shown in Figure A.1. We observe that depending on the initialization some methods can fail to converge but *extrapolation* (3.12) seem to perform better than the rest of the methods.

Model	WGAN-GP (DCGAN)	
Method	no averaging	uniform avg
SimAdam	*6.00 ± .07*	6.01 ± .08
AltAdam5	*6.25 ± .05*	**6.51 ± .05**
ExtraAdam	6.22 ± .04	6.35 ± .05
PastExtraAdam	6.27 ± 0.06	6.23 ± 0.13
OptimAdam	-	-

Table A.1: Best inception scores (averaged over 5 runs) achieved on CIFAR10 for every considered Adam variant. OptimAdam is the related *Optimistic Adam* [Daskalakis et al., 2018] algorithm. We see that the techniques of extrapolation and averaging consistently enable improvements over the baselines (in italic).

7.2 DCGAN with WGAN-GP objective

We also trained the DCGAN architecture of the WGAN experiments but this time with the WGAN-GP objective. The results are shown in Table A.1. The best results are achieved with *uniform averaging* of AltAdam5, However its iterations require to update the discriminator 5 times for every generator update. With a small drop in best final score, ExtraAdam can train WGAN-GP significantly faster (see Fig. A.2 right) as the discriminator and generator are updated only twice.

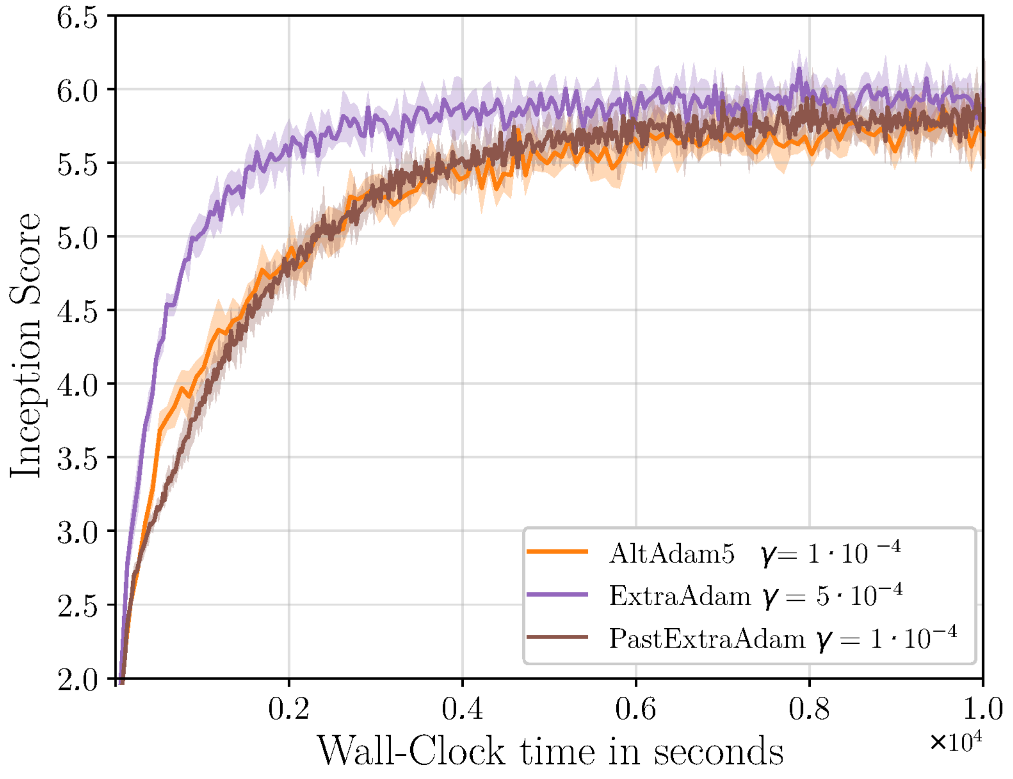
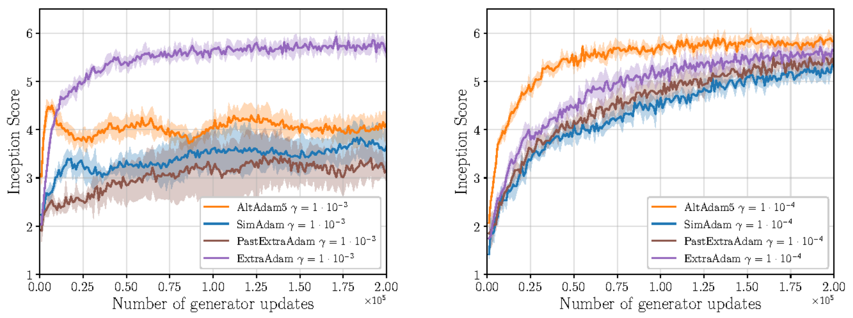


Figure A.2: DCGAN architecture with WGAN-GP trained on CIFAR10: mean and standard deviation of the inception score computed over 5 runs for each method using the best performing learning rate plotted over wall-clock time; all experiments were run on a NVIDIA Quadro GP100 GPU. We see that ExtraAdam converges faster than the Adam baselines.



(a) learning rate of 10^{-3} **(b)** learning rate of 10^{-4}

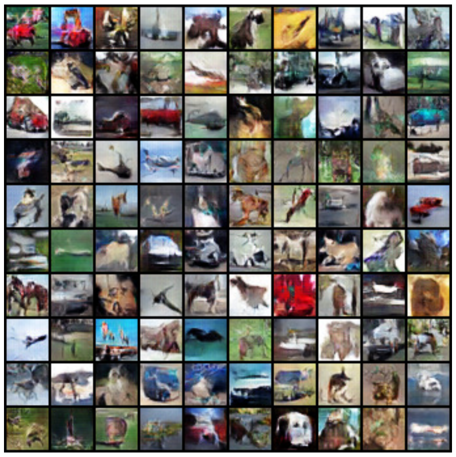
Figure A.3: Inception score on CIFAR10 for WGAN-GP over number of generator updates for different learning rates. We can see that AvgExtraSGD is less sensitive to the choice of learning rate.

7.3 Comparison of the methods with the same learning rate

In this section we compare how the methods presented in §7 perform with the same step-size. We follow the same protocol as in the experimental section §7. In Figure A.3 we plot the inception score provided by each training method as a function of the number of generator updates. Note that these plots advantage **AltAdam5** a bit because each iteration of this algorithm is a bit more costly (since it perform 5 discriminator updates for each generator update). Nevertheless, the goal of this experiment is not to show that **AltAdam5** is faster but to show that that **ExtraAdam** is less sensitive to the choice of learning rate and can be used with higher learning rates with less degradation.

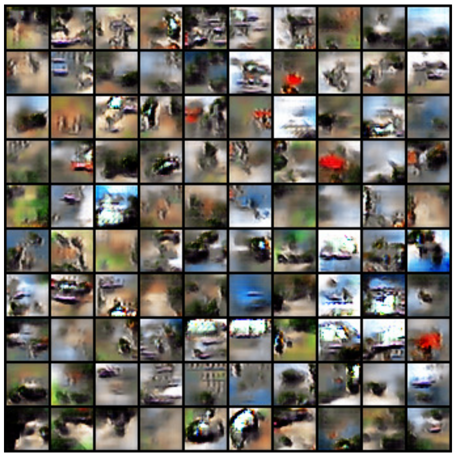
The same way in Figure A.4 we compare the sample quality on the WGAN-GP experiment of **AltAdam5** and **AvgExtraAdam** for different step-sizes. We notice that for **AvgExtraAdam** the sample quality do not significantly change whereas the sample quality of **AltAdam5** seems to be really sensitive to step-size tunning.

We think that robustness to step-size tuning is a key property for an optimization algorithm in order to save as much time as possible to tune other hyperparameters of the learning procedure such as regularization.

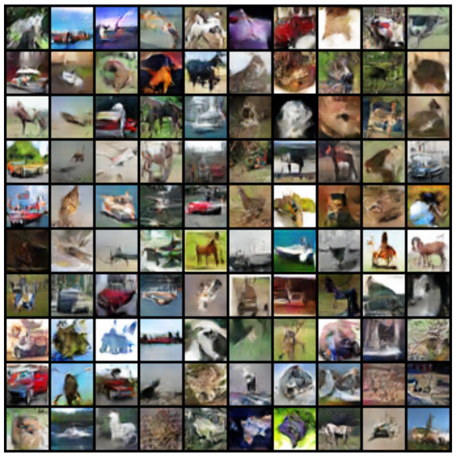


(a) AvgExtraAdam with $\eta = 10^{-3}$

(b) AvgExtraAdam with $\eta = 10^{-4}$

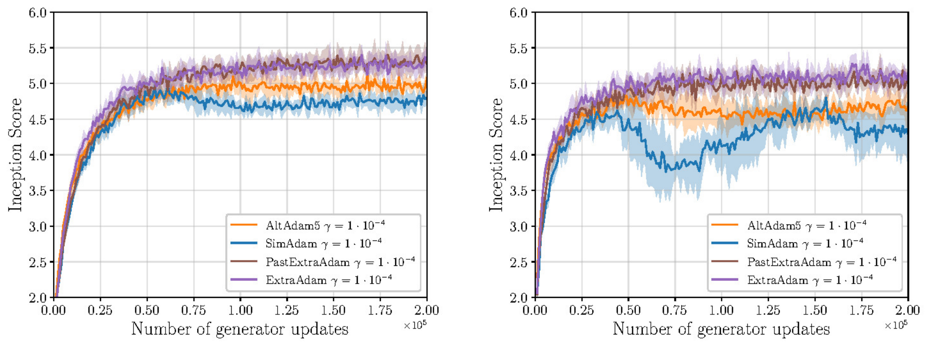


(c) AltAdam with $\eta = 10^{-3}$



(d) AltAdam with $\eta = 10^{-4}$

Figure A.4: Comparison of the samples quality on the WGAN-GP experiment for different methods and learning rate η.



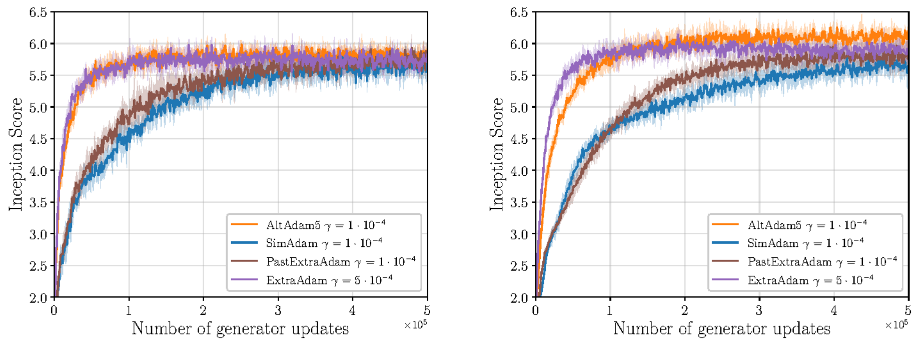
(a) with averaging **(b)** without averaging

Figure A.5: Inception Score on CIFAR10 for WGAN over number of generator updates with and without averaging. We can see that averaging improve the inception score.

7.4 Comparison of the methods with and without averaging

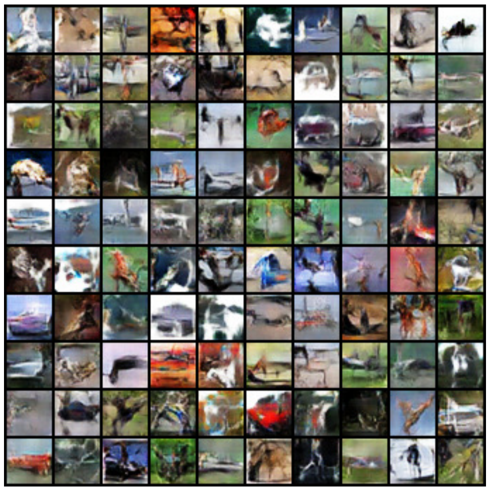
In this section we compare how uniform averaging improve the performance of the methods presented in §7. We follow the same protocol as in the experimental section §7. In Figure A.5 and A.6, we plot the inception score provided by each training method as a function of the number of generator updates with and without averaging.

We notice that uniform averaging seems to improve the inception score, nevertheless it looks like the sample are a bit more blurry (see Figure A.7) this is confirmed by our result (Figure A.8) on the Fréchet Inception Distance (FID) which is more sensitive to blurriness.

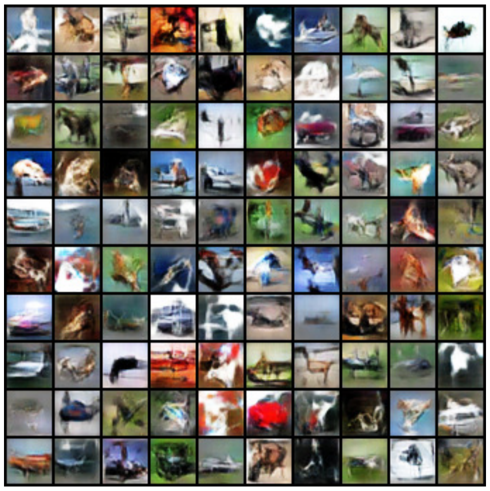


(a) with averaging **(b)** without averaging

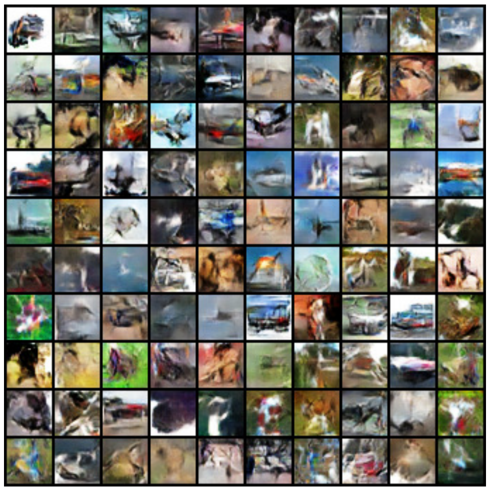
Figure A.6: Inception score on CIFAR10 for WGAN-GP over number of generator updates



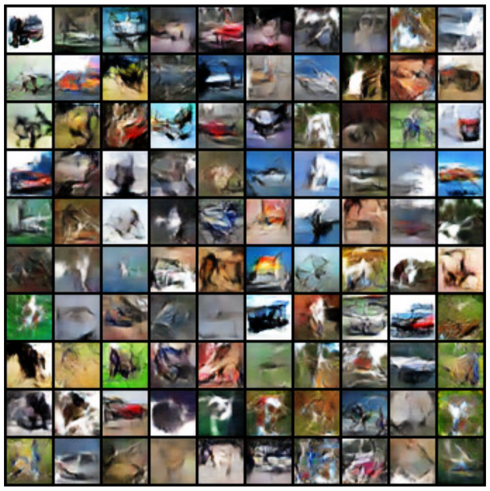
(a) AvgPastExtraSGD without averaging



(b) AvgPastExtraSGD with averaging



(c) AltSGD without averaging



(d) AltSGD with averaging

Figure A.7: Comparison of the samples of a WGAN trained with the different methods with and without averaging. Although averaging improves the inception score, the samples seem more blurry

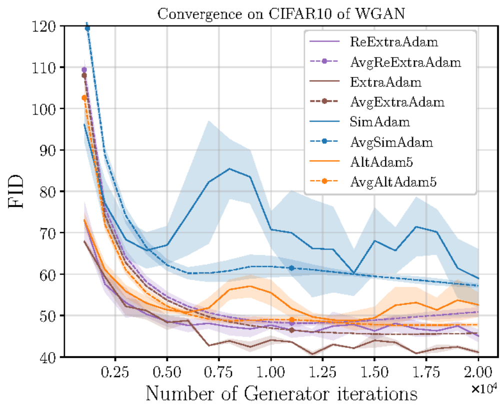


Figure A.8: The Fréchet Inception Distance (FID) from Heusel et al. [2017] computed using 50,000 samples, on the WGAN experiments. ReExtraAdam refers to Alg. 5 introduced in §4. We can see that averaging performs worse than when comparing with the Inception Score. We observed that the samples generated by using averaging are a little more blurry and that the FID is more sensitive to blurriness, thus providing an explanation for this observation.

B A Closer Look at the Optimization Landscapes of Generative Adversarial Networks

1 Proof of theorems and propositions

1.1 Proof of Theorem 1

Let us recall the theorem of interest:

Proposition 1. *Let us assume that (3.6) is an equality and that $\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$ is diagonalizable, then there exists a basis $\boldsymbol{P}$ such that the coordinates $\tilde{\boldsymbol{\omega}}(t) := \boldsymbol{P}(\boldsymbol{\omega}(t) - \boldsymbol{\omega}^*)$ have the following behavior,*

1. *For $\lambda_j \in \operatorname{Sp} \nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$, $\lambda_j \in \mathbb{R}$, we observe pure attraction:* $\tilde{\boldsymbol{\omega}}_j(t) = e^{-\lambda_j t}[\tilde{\boldsymbol{\omega}}_j(0)$.
2. *For $\lambda_j \in \operatorname{Sp} \nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$, $\Re(\lambda_j) = 0$, we observe pure rotation:* $\begin{bmatrix} \tilde{\boldsymbol{\omega}}_j(t) \\ \tilde{\boldsymbol{\omega}}_{j+1}(t) \end{bmatrix} = R_{|\lambda_j| t} \begin{bmatrix} \tilde{\boldsymbol{\omega}}_j(0) \\ \tilde{\boldsymbol{\omega}}_{j+1}(0) \end{bmatrix}$.
3. *Otherwise, we observe both:* $\begin{bmatrix} \tilde{\boldsymbol{\omega}}_j(t) \\ \tilde{\boldsymbol{\omega}}_{j+1}(t) \end{bmatrix} = e^{-\operatorname{Re}(\lambda_j)t} R_{\operatorname{Im}(\lambda_j)t} \begin{bmatrix} \tilde{\boldsymbol{\omega}}_j(0) \\ \tilde{\boldsymbol{\omega}}_{j+1}(0) \end{bmatrix}$.

The matrix R_φ corresponds to a rotation of angle φ. Note that, we re-ordered the eigenvalues such that the complex conjugate eigenvalues form pairs: if $\lambda_j \notin \mathbb{R}$ then $\lambda_{j+1} = \bar{\lambda}_j$.

Proof. The ODE we consider is,

$$\frac{d\boldsymbol{\omega}(t)}{dt} = -\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)(\boldsymbol{\omega}(t) - \boldsymbol{\omega}^*) \tag{1.1}$$

The solution of this ODE is

$$\boldsymbol{\omega}(t) = e^{-(t-t_0)\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)}(\boldsymbol{\omega}(t_0) - \boldsymbol{\omega}^*) + \boldsymbol{\omega}^* \tag{1.2}$$

Let us now consider λ an eigenvalue of $\operatorname{Sp}(\nabla \boldsymbol{v}(\boldsymbol{\omega}^*))$ such that $\operatorname{Re}(\lambda) > 0$ and $\operatorname{Im}(\lambda) \neq 0$. Since $\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$ is a real matrix and $\operatorname{Im}(\lambda) \neq 0$ we know that the complex conjugate $\bar{\lambda}$ of λ belongs to $\mathrm{Sp}(\nabla \boldsymbol{v}(\boldsymbol{\omega}^*))$. Let $\boldsymbol{u}_0$ be a complex eigenvector of λ, then we have that,

$$\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)\boldsymbol{u}_0 = \lambda \boldsymbol{u}_0 \quad \Rightarrow \quad \nabla \boldsymbol{v}(\boldsymbol{\omega}^*)\bar{\boldsymbol{u}}_0 = \bar{\lambda}\bar{\boldsymbol{u}}_0 \tag{1.3}$$

and thus $\bar{\boldsymbol{u}}_0$ is a eigenvector of $\bar{\lambda}$. Now if we set $\boldsymbol{u}_1 := \boldsymbol{u}_0 + \bar{\boldsymbol{u}}_0$ and $i\boldsymbol{u}_2 := \boldsymbol{u}_0 - \bar{\boldsymbol{u}}_0$, we have that

$$e^{-t\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)}\boldsymbol{u}_1 = e^{-t\lambda}\boldsymbol{u}_0 + e^{-t\bar{\lambda}}\bar{\boldsymbol{u}}_0 = \mathrm{Re}(e^{-t\lambda})\boldsymbol{u}_1 + \mathrm{Im}(e^{-t\lambda})\boldsymbol{u}_2 \tag{1.4}$$

$$e^{-t\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)}i\boldsymbol{u}_2 = e^{-t\lambda}\boldsymbol{u}_0 - e^{-t\bar{\lambda}}\bar{\boldsymbol{u}}_0 = i(\mathrm{Re}(e^{-t\lambda})\boldsymbol{u}_2 - \mathrm{Im}(e^{-t\lambda})\boldsymbol{u}_1) \tag{1.5}$$

Thus if we consider the basis that diagonalizes $\nabla \boldsymbol{v}(\boldsymbol{\omega}^*)$ and modify the complex conjugate eigenvalues in the way we described right after 1.3 we get the expected diagonal form in a real basis. Thus there exists $\boldsymbol{P}$ such that

$$\nabla \boldsymbol{v}(\boldsymbol{\omega}^*) = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1} \tag{1.6}$$

where $\boldsymbol{D}$ is the block diagonal matrix with the block described in Theorem 1. □

1.2 Being a DNE is neither necessary or sufficient for being a LSSP

Let us first recall Example 1.

Example 1. *Let us consider $\mathcal{L}_G$ as a hyperbolic paraboloid (a.k.a., saddle point function) centered in $(1,1)$ where $(1,\varphi)$ is the principal descent direction and $(-\varphi,1)$ is the principal ascent direction, while $\mathcal{L}_D$ is a simple bilinear objective.*

$$\mathcal{L}_G(\theta_1,\theta_2,\varphi) = (\theta_2 - \varphi\theta_1 - 1)^2 - \tfrac{1}{2}(\theta_1 + \varphi\theta_2 - 1)^2\,, \quad \mathcal{L}_D(\theta_1,\theta_2,\varphi) = \varphi(5\theta_1 + 4\theta_2 - 9)$$

We want to show that $(1,1,0)$ is a locally stable stationary point.

Proof. The game vector field has the following form,

$$\boldsymbol{v}(\theta_1,\theta_2,\varphi) = \begin{pmatrix} (2\varphi^2 - 1)\theta_1 - 3\varphi\theta_2 + 2\varphi + 1 \\ (2-\varphi^2)\theta_2 - 3\varphi\theta_1 - 2 + \varphi \\ 5\theta_1 + 4\theta_2 - 9 \end{pmatrix} \tag{1.7}$$

Thus, $(\theta_1^*,\theta_2^*,\varphi^*) := (1,1,0)$ is a stationary point (i.e., $\boldsymbol{v}(\theta_1^*,\theta_2^*,\varphi^*) = 0$). The Jacobian of the game vector field is

$$\nabla \boldsymbol{v}(\theta_1,\theta_2,\varphi) = \begin{pmatrix} 2\varphi^2 - 1 & -3\varphi & 2 - 3\theta_2 \\ -3\varphi & 2 - \varphi^2 & 1 - 3\theta_1 \\ 5 & 4 & 0 \end{pmatrix}, \tag{1.8}$$

and thus,

$$\nabla \boldsymbol{v}(\theta_1^*, \theta_2^*, \varphi^*) = \begin{pmatrix} -1 & 0 & -1 \\ 0 & 2 & -2 \\ 5 & 4 & 0 \end{pmatrix} . \tag{1.9}$$

We can verify that the eigenvalues of this matrix have a positive real part with any solver (the eigenvalues of a 3×3 always have a closed form) . For completeness we provide a proof without using the closed form of the eigenvalues. The eigenvalues $\nabla \boldsymbol{v}(\theta_1^*, \theta_2^*, \varphi^*)$ are given by the roots of its characteristic polynomial,

$$\chi(X) := \begin{vmatrix} X+1 & 0 & 1 \\ 0 & X-2 & 2 \\ -5 & -4 & 0 \end{vmatrix} = X^3 - X^2 + 11X - 2 \,. \tag{1.10}$$

This polynomial has a real root in $(0, 1)$ because $\chi(0) = -2 < 0 < 9 = \chi(1)$. Thus we know that, there exists $\alpha \in (0, 1)$ such that,

$$X^3 - X^2 + 11X - 2 = (X - \alpha)(X - \lambda_1)(X - \lambda_2) \,. \tag{1.11}$$

Then we have the equalities,

$$\alpha \lambda_1 \lambda_2 = 2 \tag{1.12}$$

$$\alpha + \lambda_1 + \lambda_2 = 1 \,. \tag{1.13}$$

Thus, since $0 < \alpha < 1$, we have that,

- If λ_1 and λ_2 are real, they have the same sign $\lambda_1 \lambda_2 = 2/\alpha > 0$) and thus are positive ($\lambda_1 + \lambda_2 = 1 - \alpha > 0$).
- If λ_1 is complex then $\lambda_2 = \bar{\lambda}_1$ and thus, $2\Re(\lambda_1) = \lambda_1 + \lambda_2 = 1 - \alpha > 0$.

□

Example 1 showed that LSSP did not imply DNE. Let us construct an example where a game have a DNE which is not locally stable.

Example 2. *Consider the non-zero-sum game with the following respective losses for each player,*

$$\mathcal{L}_1(\theta, \phi) = 4\theta^2 + (\tfrac{1}{2}\phi^2 - 1) \cdot \theta \quad \textit{and} \quad \mathcal{L}_2(\theta, \phi) = (4\theta - 1)\phi + \tfrac{1}{6}\theta^3 \tag{1.14}$$

This game has two stationary points for $\theta = 0$ and $\phi = \pm 1$. The Jacobian of the dynamics at these two points are

$$\nabla \boldsymbol{v}(0, 1) = \begin{pmatrix} 1 & 1/2 \\ 2 & 1/2 \end{pmatrix} \quad \text{and} \quad \nabla \boldsymbol{v}(0, -1) = \begin{pmatrix} 1 & -1/2 \\ 2 & -1/2 \end{pmatrix} \tag{1.15}$$

Thus,

- The stationary point $(0,1)$ is a DNE but $\mathrm{Sp}(\nabla \boldsymbol{v}(0,1)) = \{\frac{3\pm\sqrt{17}}{4}\}$ contains an eigenvalue with negative real part and so is *not* a LSSP.
- The statioanry point $(0,-1)$ is *not* a DNE but $\mathrm{Sp}(\nabla \boldsymbol{v}(0,1)) = \{\frac{1\pm i\sqrt{7}}{4}\}$ contains only eigenvalue with positive real part and so is a LSSP.

2 Computation of the top-k Eigenvalues of the Jacobian

Neural networks usually have a large number of parameters, this usually makes the storing of the full Jacobian matrix impossible. However the Jacobian vector product can be efficiently computed by using the trick from [Pearlmutter, 1994]. Indeed it's easy to show that $\nabla \boldsymbol{v}(\boldsymbol{\omega})\boldsymbol{u} = \nabla(\boldsymbol{v}(\boldsymbol{\omega})^T \boldsymbol{u})$.

To compute the eigenvalues of the Jacobian of the Game, we first compute the gradient $\boldsymbol{v}(\boldsymbol{\omega})$ over a subset of the dataset. We then define a function that computes the Jacobian vector product using automatic differentiation. We can then use this function to compute the top-k eigenvalues of the Jacobian using the `sparse.linalg.eigs` functions of the Scipy library.

3 Experimental Details

3.1 Mixture of Gaussian Experiment

Dataset. The Mixture of Gaussian dataset is composed of 10,000 points sampled independently from the following distribution $p_{\mathcal{D}}(x) = \frac{1}{2}\mathcal{N}(2, 0.5) + \frac{1}{2}\mathcal{N}(-2, 1)$ where $\mathcal{N}(\mu, \sigma^2)$ is the probability density function of a 1D-Gaussian distribution with mean μ and variance σ^2. The latent variables $z \in \mathbb{R}^d$ are sampled from a standard Normal distribution $\mathcal{N}(0, I_d)$. Because we want to use full-batch methods, we sample 10,000 points that we re-use for each iteration during training.

Neural Networks Architecture. Both the generator and discriminator are one hidden layer neural networks with 100 hidden units and ReLU activations.

WGAN Clipping. Because of the clipping of the discriminator parameters some components of the gradient of the discriminator's gradient should no be taken into account. In order to compute the relevant path angle we apply the following filter to the gradient:

$$\mathbf{1}\left\{(|\boldsymbol{\varphi}| = \mathbf{c}) \ \texttt{and} \ (\mathrm{sign}\nabla_{\boldsymbol{\varphi}}\mathcal{L}_{\mathbf{D}}(\boldsymbol{\omega}) = -\mathrm{sign}\boldsymbol{\varphi})\right\} \tag{3.1}$$

where $\boldsymbol{\varphi}$ is clipped between $-c$ and c. If this condition holds for a coordinate of the gradient then it mean that after a gradient step followed by a clipping the value of the coordinate will not change.

Hyperparameters for WGAN-GP on MoG	
Batch size	$= 10,000$ (Full-Batch)
Number of iterations	$= 30,000$
Learning rate for generator	$= 1 \times 10^{-2}$
Learning rate for discriminator	$= 1 \times 10^{-1}$
Gradient Penalty coefficient	$= 1 \times 10^{-3}$

Hyperparameters for NSGAN on MoG	
Batch size	$= 10,000$ (Full-Batch)
Number of iterations	$= 30,000$
Learning rate for generator	$= 1 \times 10^{-1}$
Learning rate for discriminator	$= 1 \times 10^{-1}$

3.2 MNIST Experiment

Dataset We use the training part of MNIST dataset LeCun et al. [2010] (50K examples) for training our models, and scale each image to the range $[-1, 1]$.

Architecture We use the DCGAN architecture Radford et al. [2016] for our generator and discriminator, with both the NSGAN and WGAN-GP objectives. The only change we make is that we replace the Batch-norm layer in the discriminator with a Spectral-norm layer Miyato et al. [2018], which we find to stabilize training.

Training Details

Hyperparameters for NSGAN with Adam	
Batch size	$= 100$
Number of iterations	$= 100,000$
Learning rate for generator	$= 2 \times 10^{-4}$
Learning rate for discriminator	$= 5 \times 10^{-5}$
β_1	$= 0.5$

Hyperparameters for NSGAN with ExtraAdam	
Batch size	$= 100$
Number of iterations	$= 100,000$
Learning rate for generator	$= 2 \times 10^{-4}$
Learning rate for discriminator	$= 5 \times 10^{-5}$
β_1	$= 0.9$

Hyperparameters for WGAN-GP with Adam	
Batch size	$= 100$
Number of iterations	$= 200,000$
Learning rate for generator	$= 8.6 \times 10^{-5}$
Learning rate for discriminator	$= 8.6 \times 10^{-5}$
β_1	$= 0.5$
Gradient penalty λ	$= 10$
Critic per Gen. iterations λ	$= 5$

Hyperparameters for WGAN-GP with ExtraAdam	
Batch size	$= 100$
Number of iterations	$= 200,000$
Learning rate for generator	$= 8.6 \times 10^{-5}$
Learning rate for discriminator	$= 8.6 \times 10^{-5}$
β_1	$= 0.9$
Gradient penalty λ	$= 10$
Critic per Gen. iterations λ	$= 5$

Computing Inception Score on MNIST. We compute the inception score (IS) for our models using a LeNet classifier pretrained on MNIST. The average IS score of real MNIST data is 9.9.

3.3 Path-Angle Plot

We use the path-angle plot to illustrate the dynamics close to a LSSP. To compute this plot, we need to choose an initial point $\boldsymbol{\omega}$ and an end point $\boldsymbol{\omega}'$. We choose the $\boldsymbol{\omega}$ to be the parameters at initialization, but $\boldsymbol{\omega}'$ can more subtle to choose. In practice, when we use stochastic gradient methods we typically reach a neighborhood of a LSSP where the norm of the gradient is small. However, due to the stochastic noise, we keep moving around the LSSP. In order to be robust to the choice of the end point ω', we take multiple close-by points during training that have good performance (e.g., high IS in MNIST). In all of figures, we compute the path-angle (and path-norm) for all these end points (with the same start point), and we plot the median path-angle (middle line) and interquartile range (shaded area).

3.4 Instability of Gradient Descent

For the MoG dataset we tried both the extragradient method [Korpelevich, 1976, Gidel et al., 2019a] and the standard gradient descent. We observed that gradient descent leads to unstable results. In particular the norm of the gradient has very large variance compared to extragradient this is shown in Fig. B.1.

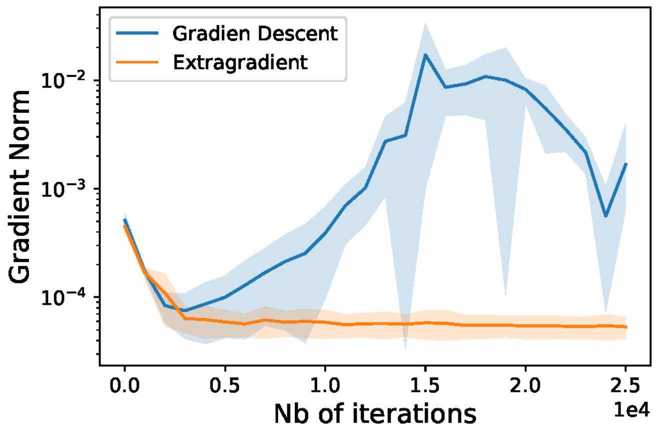
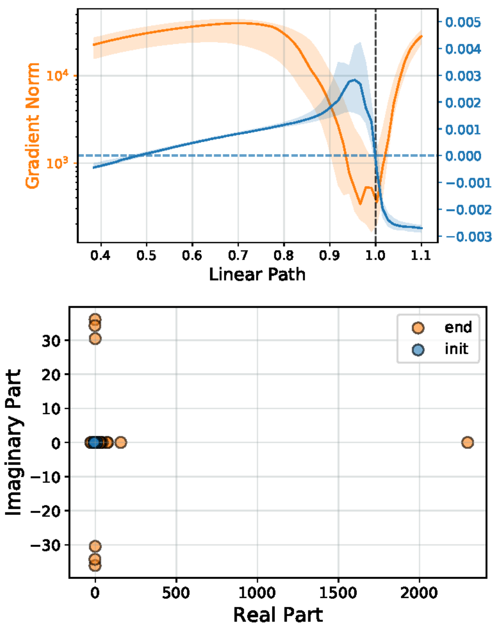


Figure B.1: The norm of gradient during training for the standard GAN objective. We observe that while extra-gradient reaches low norm which indicates that it has converged, the gradient descent on the contrary doesn't seem to converge.

3.5 Additional Results with Adam


(a) NSGAN on MNIST, IS: 8.95

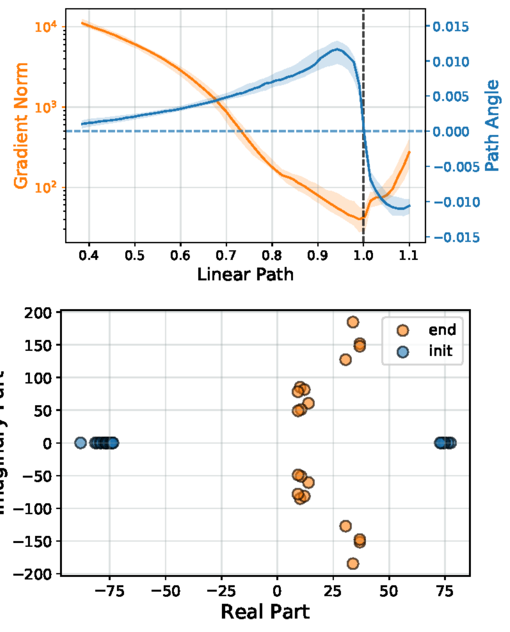

(b) WGAN-GP on MNIST, IS: 9.30

Figure B.2: Path-angle and Eigenvalues computed on MNIST with Adam.

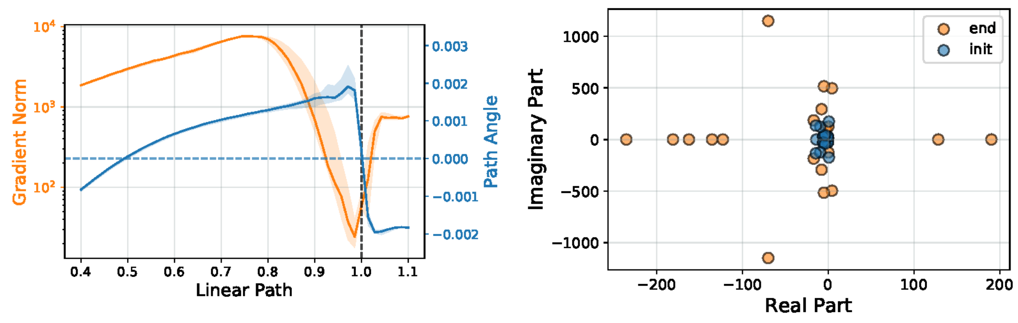


Figure B.3: Path-angle and Eigenvalues for NSGAN on CIFAR10 computed on CIFAR10 with Adam. We can see that the model has eigenvalues with negative real part, this means that we've actually reached an unstable point.

C Adversarial Example Games

1 Proofs for Section 4 (Theoretical results)

Proof. Let us recall that the payoff φ is defined as

$$\max_{g\in\mathcal{G}_\epsilon}\min_{f\in\mathcal{F}}\mathbb{E}_{(x,y)\sim p_{data},z\sim p_z}[\ell(f(g(x,y,z)),y)] =: \varphi(f,g) \tag{1.1}$$

Let us consider the case where $\mathcal{X}$ is finite as a warm-up. In practice, this can be the case if we consider that for instance one only allows a finite number of values for the pixels, e.g. (integers between 0 and 255 for CIFAR-10). In that case, we have that

$$\mathbb{P}_{adv}(x,y) = \sum_{x'\in\mathcal{X}}\mathbb{P}_{data}(x',y)\mathbb{P}_g(x|x',y) \quad \text{where} \quad d(x,x') > \epsilon \Rightarrow \mathbb{P}_g(x|x',y) = 0\,. \tag{1.2}$$

Assuming that one can achieve any $\mathbb{P}_g(\cdot|x',y)$ respecting the proximity constraint, the set $\{\mathbb{P}_{adv}\}$ is convex and compact. It is compact because closed and bounded in finite dimension and convex because of the linear dependence in $\mathbb{P}_g$ in (1.2) (and the fact that if $\mathbb{P}_{g_1}$ and $\mathbb{P}_{g_2}$ respect the constraints then $\lambda\mathbb{P}_{g_1}+(1-\lambda)\mathbb{P}_{g_2}$ does it too).

For the non finite input case, we can consider that the generator is a random variable defined on the probability space $(\mathcal{X}\times\mathcal{Y}\times\mathcal{Z},\mathcal{B},\mathbb{P}_{(x,y)}\times\mathbb{P}_z := \mathbb{P})$ where $\mathcal{B}$ is the Borel σ-algebra, $\mathbb{P}_{(x,y)}$ the probability on the space of data and $\mathbb{P}_z$ the probability on the latent space. Then the adversarial distributions p_g we consider is the pushforward distributions $\mathbb{P}\circ G^{-1}$ with G such that,

$$G(x,y,z) = (g(x,y,z),y) \quad \text{and} \quad d(g(x,y,z),x) \le \epsilon\,,\ \forall x,y,z\in\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}\,. \tag{1.3}$$

It implies that $\Delta_\epsilon := \{p_g \,:\, g\in\mathcal{G}_\epsilon\} = \{\mathbb{P}\circ G^{-1}\,|\,G \text{ measurable satisfying } (1.3)\}$. By definition of the payoff φ and (3.1), we have that

$$\varphi(f_c,g) := \mathbb{E}_{(x,y)\sim\mathcal{D},z\sim p_z}[\ell(f_c(g(x,y,z)),y)] = \mathbb{E}_{(x',y)\sim p_g}[\ell(f_c(x'),y)] =: \varphi(f_c,p_g)$$

and thus it lead to the equivalence,

$$\min_{f_c\in\mathcal{F}}\max_{p_g\in\Delta_\epsilon}\varphi_\lambda(f_c,p_g) = \max_{p_g\in\Delta_\epsilon}\min_{f_c\in\mathcal{F}}\varphi_\lambda(f_c,p_g) \iff \min_{f_c\in\mathcal{F}}\max_{g\in\mathcal{G}_\epsilon}\varphi_\lambda(f_c,g) = \max_{g\in\mathcal{G}_\epsilon}\min_{f_c\in\mathcal{F}}\varphi_\lambda(f_c,g)$$

Recall that we assumed that one has access to any measurable g that satisfies (1.3), the minimax problem (4.1). Let us show that under this assumption the set Δ_ϵ is convex and compact.

Δ_ϵ is convex: let us consider $\mathbb{P} \circ G_1^{-1}$ and $\mathbb{P} \circ G_2^{-1}$ we have that

$$\lambda \mathbb{P} \circ G_1^{-1} + (1-\lambda)\mathbb{P} \circ G_2^{-1} = \mathbb{P} \circ G_3^{-1} \tag{1.4}$$

where $g_3(x,y,z) = \delta(z) g_1(x,y,z) + (1-\delta(z)) g_2(x,y,z)$ and where $\delta \sim Ber(\lambda)$. Note that g_3 satisfies (1.3) by convexity of $x \mapsto d(x,x')$.

Δ_ϵ is compact: By using Skorokhod's representation theorem [Billingsley, 1999] we can show that this set is closed and thus compact (as closed subsets of compact sets are compact) using the weak convergence of measures as topology.

Thus Δ_ϵ is a convex compact Haussdorf space and in both cases ($\mathcal{X}$ finite and infinite) and we can apply Fan's Theorem.

Theorem 2. *[Fan, 1953, Theorem 2] Let U be a compact and convex Hausdorff space and V an arbitrary convex set. Let φ be a real valued function on $U \times V$ such that for every $v \in V$ the function $\varphi(\cdot, v)$ is lower semi-continuous on U. If φ is convex-concave then,*

$$\min_{u \in U} \sup_{v \in V} \varphi(u,v) = \sup_{v \in V} \min_{u \in U} \varphi(u,v) \tag{1.5}$$

Note that we do not prove this result for neural networks. □

Proof. We prove the result here for any given norm $\|\cdot\|$. Let us consider the loss for a given pair (x,y)

$$\log(1 + e^{y(w^\top g(x,y)+b)}) \tag{1.6}$$

then by the fact that $x \mapsto \log(1+e^x)$ is increasing, maximizing this term for $\|g(x,y) - x\|_\infty \leq \epsilon$, boils down to solving the following maximization step,

$$\max_{\delta\,,\, \|\delta\| \leq \epsilon} y(w^\top \delta) = \|w\|_* \tag{1.7}$$

Particularly, for the ℓ_∞ norm we get

$$\arg \max_{\delta\,,\, \|\delta\|_\infty \leq \epsilon} y(w^\top \delta) = \epsilon y \, \mathrm{sign}(w)\,. \tag{1.8}$$

and

$$\max_g \mathbb{E}_{(x,y)\sim p_{data}}[\log(1 + e^{y(w^\top g(x,y)+b)})] = \mathbb{E}_{(x,y)\sim p_{data}}[\log(1 + e^{y(w^\top x+b)+\epsilon\|w\|_1})] \tag{1.9}$$

To show that (f^*, g^*) is a Nash equilibrium of the game (4.2), we first notice that by construction

$$\varphi(f^*, g^*) = \min_{f \in \mathcal{F}} \max_{g} \varphi(f, g) \tag{1.10}$$

where $\mathcal{F}$ is the class of classifier with linear logits. We then just need to notice that for all $f \in \mathcal{F}$ we have,

$$\varphi(f, g^*) = \mathbb{E}_{(x,y)\sim p_{data}}[\log(1 + e^{-y(w^\top x + b) + \epsilon w^\top \operatorname{sign}(w^*)})] \tag{1.11}$$

That is a convex problem in (w, b). Thus by assuming that w^* is full support and since w^* minimize (4.3) we have that

$$\nabla_w \varphi(w^*, b^*, g^*) = \nabla_w \mathbb{E}_{(x,y)\sim p_{data}}[\log(1 + e^{-y(w^\top x + b) + \epsilon \|w\|_1})] = 0 \tag{1.12}$$

Finally, by convexity of the problem (1.11) we can conclude that w^* is a minimizer of $\varphi(\cdot, g^*)$. To sum-up we have that

$$\varphi(f^*, g) \leq \varphi(f^*, g^*) \leq \varphi(f, g^*) \tag{1.13}$$

meaning that (f^*, g^*) is a Nash equilibrium of (4.2). □

Proof. In the case where $f^*(x) := p(y|x) \in \mathcal{F}$, since f^* a minimizer of the expected cross entropy loss over the class of any function, we have that

$$H_y(p) := \min_{f \in \mathcal{X}^{\mathcal{Y}}} \mathbb{E}_{(x,y)\sim p}[\ell(f(x), y)] = \mathbb{E}_x[H(p(\cdot|x))] \tag{1.14}$$

Lemma 9. *Given a data distribution $(x, y) \sim p_{adv}$ a minimizer of the cross entropy loss is*

$$p_{adv}(y|\cdot) \in \arg\min_f \mathbb{E}_{(x,y)\sim p_{adv}}[\ell(f(x), y)] \,. \tag{1.15}$$

Proof. Let us start by noticing that,

$$\min_f \mathbb{E}_{(x,y)\sim p}[\ell(f(x), y)] = \mathbb{E}_{x\sim p_x} \min_{q = f(x)} \mathbb{E}_{y\sim p(\cdot|x)}[\ell(q, y)] \tag{1.16}$$

Using the fact that ℓ is the cross-entropy loss we get

$$\mathbb{E}_{y\sim p(\cdot|x)}[\ell(q, y)] = \mathbb{E}_{y\sim p(\cdot|x)}[-\sum_{i=1}^{K} y_i \ln(q_i)] = -\sum_{i=1}^{K} p_i \ln(q_i) \tag{1.17}$$

where we noted $p_i = p(y = i|x)$. Noticing that since q_i is a probability distribution we have $\sum_{i=1}^{K} q_i = 1$, we have,

$$\mathbb{E}_{y\sim p(\cdot|x)}[\ell(q, y)] = -\sum_{i=1}^{K-1} p_i \ln(q_i) - p_K \ln(1 - \sum_{i=1}^{K-1} q_i) \tag{1.18}$$

we can then differentiate this loss with respect to $q_i \geq 0$ and get,

$$\frac{\partial \mathbb{E}_{y\sim p(\cdot|x)}[\ell(q,y)]}{\partial q_i}(q) = -\frac{p_i}{q_i} + \frac{p_K}{q_K} \tag{1.19}$$

We can finally notice that $q_i = p_i$ is a feasible solution. $\square$

$\square$

2 Experimental Details

The experiments are subject to different sources of variations, in all our experiments we try to take into account those sources of variations when reporting the results. We detail the different sources of variations for each experiment and how we report them in the next section.

2.1 Source of variations

NoBox attacks on a known architecture class. we created a random split of the MNIST and CIFAR10 dataset and trained a classifier on each splits. To evaluate each method we then use one of the classifiers as the source classifier and all the other classifiers as the targets we want to attack. We then compute the mean and standard deviation of the attack success rates across all target classifiers. To take into account the variability in the results that comes from using a specific classifier as the source model, we also repeat the evaluation by changing the source model. We report the average and 95% interval (assuming the results follow a normal distribution) in Table 8.1 by doing macro-averaging overall evaluations.

NoBox attacks across distinct architectures. For each architecture, we trained 10 different models. When evaluated against a specific architecture we evaluate against all models of this architecture. In Table 2, we report the mean and standard deviation of the error rates across all models.

NoBox Attacks against robust classifiers. For this experiment, we could only train a single robust target model per architecture because of our computational budget. The only source of variations is thus due to the inherent stochasticity of each method. Evaluating this source of randomness would require to run each method several times, unfortunately, this is quite expensive and our computational budget didn't allow for it. In Table 3, we thus only report a single number per architecture.

3 Additional results

3.1 Quantitative Results

We now provide additional results in the form of whitebox and blackbox query attacks adapted to the NoBox evaluation protocol for Known-Architecture attacks which is the experimental setting in **Q1**. For whitebox attacks we evaluate APGD-CE and APGD-DLR [Croce and Hein, 2020] which are improvements over the powerful PGD attack [Madry et al., 2018]. When ensembled with another powerful perturbation minimizing whitebox attack FAB [Croce and Hein, 2019] and the query efficient blackbox Square attacks [Andriushchenko et al., 2020] yields the current SOTA attack strategy called AutoAttack [Croce and Hein, 2020]. Additionally, we compare with two parametric blackbox query approaches that both utilize a latent space in AutoZoom [Tu et al., 2019] and $\mathcal{N}$Attack [Li et al., 2019]. To test transferability of whitebox and blackbox query attacks in the NoBox known architecture setting we give generous iteration and query budgets (10x the reported settings in the original papers) when attacking the source models, but only a single query for each target model. It is interesting to note that APGD variant whitebox attacks are significantly more effective than query based blackbox attacks but lack the same effectiveness of NoBox baselines. We hypothesize that the transferability of whitebox attacks may be due to the fact that different functions learn similar decision boundaries but different enough such that minimum distortion whitebox attacks such as FAB are ineffective.

3.2 Qualitative Results

As a sanity check we also provide some qualitative results about the generated adversarial attacks. In Figure C.1 we show the 256 attacked samples generated by our method on MNIST. In Figure C.2 we show on the left the 256 CIFAR samples to attack, and on the right the perturbations generated by our method amplified by a factor 10.

		MNIST	CIFAR-10
Whitebox	AutoAttack*	84.4 ± 5.1	91.0 ± 1.9
	APGD-CE	95.8 ± 1.9	97.5 ± 0.7
	APGD-DLR	83.9 ± 5.4	90.7 ± 2.1
	FAB	5.4 ± 2.2	10.4 ± 1.7
Blackbox-query	Square	60.9 ± 10.3	21.9 ± 2.8
	$\mathcal{N}$-Attack	9.5 ± 3.2	56.7 ± 8.9
Non-Interactive Blackbox	MI-Attack	93.7 ± 1.1	**99.9** ± 0.1
	DI-Attack	95.9 ± 1.6	**99.9 ± 0.1**
	TID-Attack	92.8 ± 2.7	19.7 ± 1.5
	SGM-Attack	N/A	**99.8** ± 0.3
	AEG (Ours)	**98.9 ± 1.4**	98.5 ± 0.6

Table C.1: Test error rates for average blackbox transfer over architectures at $\epsilon = 0.3$ for MNIST and $\epsilon = 0.03125$ for CIFAR-10 (higher is better)

Source	Attack	VGG-16	RN-18	WR	DN-121	Inc-V3
	Clean	11.2 ± 0.9	13.1 ± 2.0	6.8 ± 0.7	11.2 ± 1.4	9.9 ± 1.3
WR	MI-Attack	67.8 ± 3.01	86.0 ± 1.7	**99.9 ± 0.1**	89.0 ± 2.6	88.2 ± 1.4
	DI-Attack	68.3 ± 2.4	88.5 ± 2.1	**99.9 ± 0.1**	91.2 ± 1.6	**91.5 ± 1.8**
	TID-Attack	23.1 ± 1.8	25.9 ± 1.3	20.6 ± 1.0	23.6 ± 1.2	21.9 ± 1.7
	SGM-Attack	69.1 ± 2.1	88.6 ± 2.0	**99.6 ± 0.4**	90.7 ± 1.9	86.8 ± 2.2
	AEG (Ours)	40.8 ± 3.22	70.6 ± 4.9	98.5 ± 0.6	**88.2 ± 4.6**	**89.6 ± 1.8**
	AEG (New)	**86.2 ± 1.8**	**94.1 ± 1.5**	81.1 ± 1.1	**93.1 ± 1.7**	89.2 ± 1.8

Table C.2: Error rates on $\mathcal{D}$ for average NoBox architecture transfer attacks with $\epsilon = 0.03125$ with Wide-ResNet architecture

4 Implementation Details

We now provide additional details on training the representative classifiers and generators used in the AEG framework as outlined in 8.2. We solve the game using the ExtraAdam optimizer [Gidel et al., 2019a] with a learning rate of 10^{-3}. We allow the generator to update its parameters several times on the same batch of examples before updating the critic. In particular we update the generator until it is able to fool the critic or it reaches some fixed number of iterations. We set this max number of iterations to 20 in all our experiments. We also find that biasing the critic update various forms of adversarial training consistently leads to the most effective attack. We reconcile this phenomenon by noting that through adversarial training the critic itself becomes a robust model which provides a richer

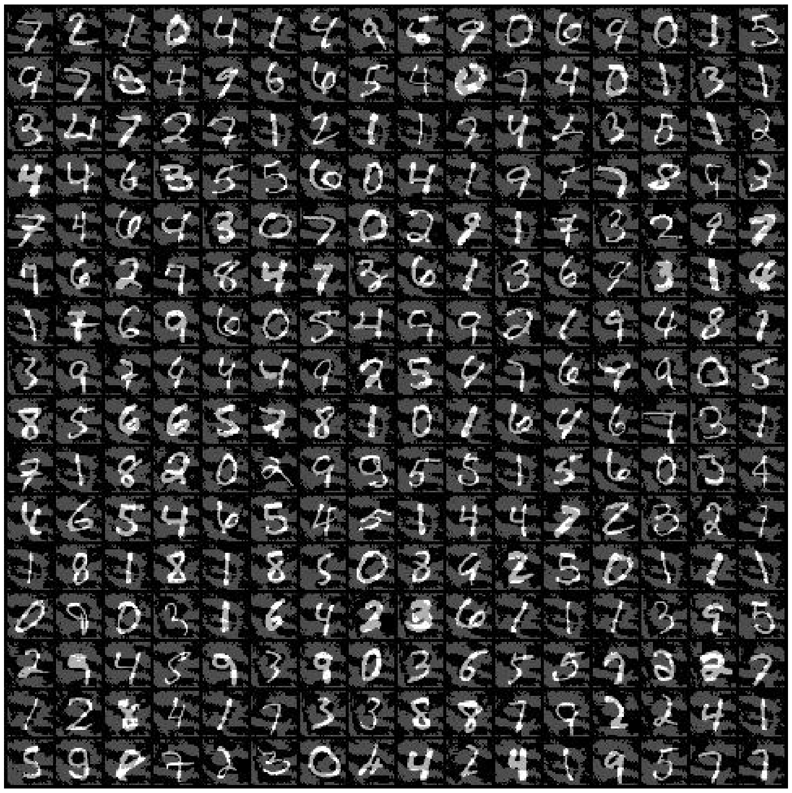


Figure C.1: Attacks generated on MNIST by our method.

learning signal to the generator. Furthermore, the elegance of the AEG frameworks allows the practitioner to further bias the optimization process of the critic —and consequently the generator— through picking and choosing effective robustness techniques such as training with PGD adversarial examples generated at a prior timestep.

4.1 Generator Architecture

The architecture we used for the encoder and the decoder is described in Table C.4 and C.5. For MNIST we used a standard convolutional architecture and for CIFAR-10 we used a ResNet architecture.

d-**ResBlock**
Input:x
Forward for computing $F(x)$:
Reflection pad (1)
conv. (ker: 3×3, $d \rightarrow d$; stride: 1; pad: 1)
Batch Normalization
ReLU
Reflection pad (1)
conv. (ker: 3×3, $d \rightarrow d$; stride: 1; pad: 1)
Batch Normalization
Output:$x + F(x)$

Table C.3: ResNet blocks used for the ResNet architectures (see Table C.4) for the Generator. Each ResNet block contains skip connection (bypass), and a sequence of convolutional layers, normalization, and the ReLU non–linearity.

Encoder
Input: $x \in \mathbb{R}^{3\times32\times32}$
Reflection Padding (3)
conv. (ker: 7×7, 32 → 63; stride: 1; pad: 0)
Batch Normalization
ReLU
conv. (ker: 3×3, 63 → 127; stride: 2; pad: 0)
Batch Normalization
ReLU
conv. (ker: 3×3, 127 → 255; stride: 2; pad: 0)
Batch Normalization
ReLU
255-ResBlock
255-ResBlock
255-ResBlock

Decoder
Input: $(\psi(x), z, y) \in \mathbb{R}^{256\times8\times8}$
256-ResBlock
256-ResBlock
256-ResBlock
Transp. conv. (ker: 3×3, 256 → 128; stride: 2; pad: 0)
Batch Normalization
ReLU
Transp. conv. (ker: 3×3, 128 → 64; stride: 2; pad: 0)
Batch Normalization
ReLU
ReflectionPadding(3)
conv. (ker: 7×7, 64 → 32; stride: 1; pad: 0)
Tanh

Table C.4: Encoder and Decoder for the convolutional generator used for the MNIST dataset.

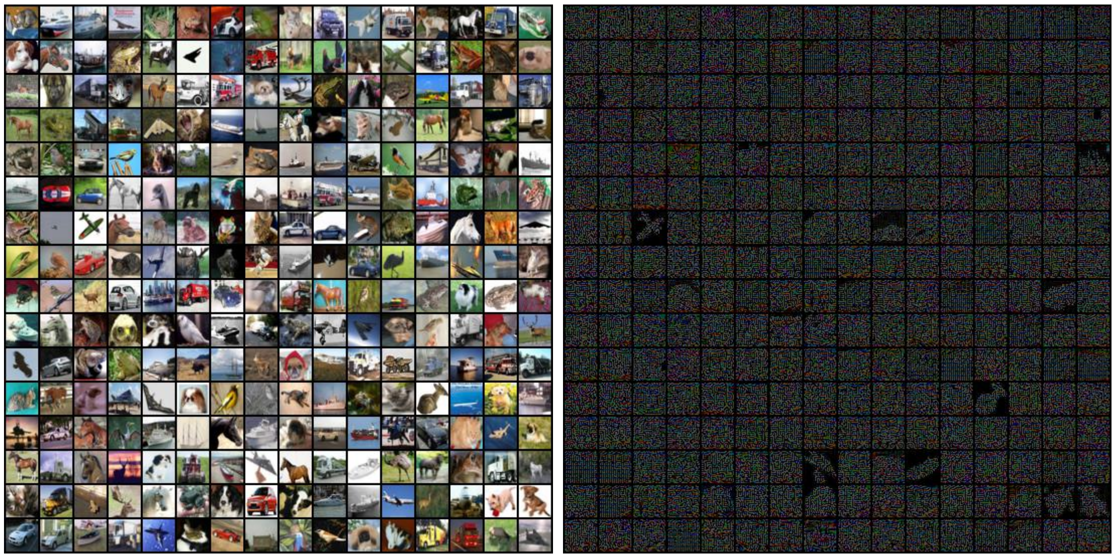


Figure C.2: Left: CIFAR examples to attack. **Right:** Pertubations generated by our method amplified by a factor 10. An interesting observation is that the generator learns not to attack the pixel where the background is white.

4.2 Baseline Implementation Details

The principal baselines used in the main paper include the Momentum-Iterative Attack (MI-Attack) [Dong et al., 2018], the Input Diversity (DI-Attack) [Xie et al., 2019], the Translation-Invariant (TID-Attack) [Dong et al., 2019] and the Skip Gradient Method (SGM-Attack) [Wu et al., 2020]. As Input Diversity and Translation invariant are approaches that generally can be combined with existing attack strategies we choose to use the powerful Momentum-Iterative attack as our base attack. Thus the DI-Attack consists of random input transformations when using an MI-Attack adversary while the TID-attack further adds a convolutional kernel ontop of the DI-Attack. We base our implementions using the AdverTorch [Ding et al., 2019] library and adapt all baselines to this framework using original implementations where available. In particular, when possible we reused open source code in the Pytorch library [Paszke et al., 2019] otherwise we re-implement existing algorithms. We also inherit most hyperparameters settings when reporting baseline results except for number steps used in iterated attacks. We find that most iterated attacks benefit from additional optimization steps when attacking MNIST and CIFAR-10 classifiers. Specifically, we allot a 100 step budget for all iterated attacks which is often a five to ten fold increase than the reported setting in all baselines.

Encoder
Input: $x \in \mathbb{R}^{28\times28}$
conv. (ker: 3×3, 1 $\rightarrow$ 64; stride: 3; pad: 1)
LeakyReLU(0.2)
Max Pooling (stride: 2)
conv. (ker: 3×3, 64 $\rightarrow$ 32; stride: 2; pad: 1)
LeakyReLU(0.2)
Max Pooling (stride: 2)

Decoder
Input: $(\psi(x), z, y) \in \mathbb{R}^{64\times2\times2}$
Transp. conv. (ker: 3×3, 64 $\rightarrow$ 32; stride: 2; pad: 1)
LeakyReLU(0.2)
Max Pooling stride 2
Reflection Padding (3)
Transp. conv. (ker: 5×5, 32 $\rightarrow$ 16; stride: 3; pad: 1)
LeakyReLU(0.2)
Max Pooling stride 2
Transp. conv. (ker: 2×2, 16 $\rightarrow$ 1; stride: 2; pad: 1)
Tanh

Table C.5: Encoder and Decoder for the ResNet generator used for the MNIST dataset.

4.3 Ensemble Adversarial Training Architectures

We ensemble adversarially train our models in accordance with the training protocol outlined in Tramèr et al. [2018]. For MNIST models we train a standard model for 6 epochs, and an ensemble adversarial model using adversarial examples from the remaining three architectures for 12 epochs. The specific architectures for Models A-D are provided in Table. 8. Similarly, for CIFAR-10 we train both the standard model and ensemble adversarial models for 50 epochs. For computationally efficiency we randomly sample two out of three held out architectures when ensemble adversarially training the source model.

A	B	C	D
Conv(64, 5, 5) + Relu	Dropout(0.2)	Conv(128, 3, 3) + Tanh	FC(300) + Relu
Conv(64, 5, 5) + Relu	Conv(64, 8, 8) + Relu	MaxPool(2,2)	Dropout(0.5)
Dropout(0.25)	Conv(128, 6, 6) + Relu	Conv(64, 3, 3) + Tanh	FC(300) + Relu
FC(128) + Relu	Conv(128, 6, 6) + Relu	MaxPool(2,2)	Dropout(0.5)
Dropout(0.5)	Dropout(0.5)	FC(128) + Relu	FC(300) + Relu
FC + Softmax	FC + Softmax	FC + Softmax	Dropout(0.5)
			FC(300) + Relu
			Dropout(0.5)
			FC + Softmax

Table C.6: MNIST Ensemble Adversarial Training Architectures)

5 Further Related Work

Adversarial attacks can be classified under different threat models, which impose different access and resource restrictions on the attacker Akhtar and Mian [2018]. The whitebox setting, where the attacker has full access to the model parameters and outputs, thus allowing the attacker to utilize gradients based methods to solve a constrained optimization procedure. This setting is more permissive than the semi-whitebox and the blackbox setting, the latter of which the attacker has only access to the prediction [Papernot et al., 2016a, 2017] or sometimes the predicted confidence [Guo et al., 2019]. In this paper, we focus on a challenging variant of the conventional blackbox threat model which we call the NoBox setting which further restricts the attacker by *not* allowing any query from the target model. While there exists a vast literature of adversarial attacks, we focus on ones that are most related to our setting and direct the interested reader to comprehensive surveys for adversarial attacks and blackbox adversarial attacks [Bhambri et al., 2019, Chakraborty et al., 2018].

Whitebox Attacks. The most common threat model for whitebox adversarial examples are l_p-norm attacks, where $p \in \{2, \infty\}$ is the choice of norm ball used to define the attack budget. One of the earliest gradient based attacks is the Fast Gradient Sign Method (FGSM) [Goodfellow et al., 2015a], which computes bounded perturbations in a single step by computing the signed gradient of the loss function with respect to a clean input. More powerful adversaries can be computed using multi-step attacks such as DeepFool [Moosavi-Dezfooli et al., 2016] which iteratively finds the minimum distance over perturbation direction needed to cross a decision boundary. For constrained optimization problems the Carlini-Wagner (CW) attack [Carlini and Wagner, 2017b] is a powerful iterative optimization scheme which introduces an attack objective designed to maximize the distance between the target class and the most likely adversarial class. Similarly, projected gradient descent based attacks has been shown to be the strongest class of adversaries for l_2 and l_∞ norm attacks [Madry et al., 2018] and even provides a natural way of robustifying models through adversarial training. Extensions of PGD that fix failures due to suboptimal step size and problems of the objective function include AutoPGD-CE (APGD-CE) and AutoPGD-DLR (APGD-DLR) and leads to the state of the art whitebox attack in AutoAttack [Croce and Hein, 2020] which ensembles two other strong diverse and parameter free attacks.

Blackbox Attacks. Like whitebox attacks the adversarial goal for a blackbox attacker remains the same with the most common threat model also being l_p norm attacks. Unlike, whitebox attacks the adversarial capabilities of the attacker is severely restricted rendering exact gradient computation impossible. In lieu of exact gradients, early blackbox attacks generated adversarial examples on surrogate models in combination with queries to the target model [Papernot et al., 2016a]. When given a query budget gradient estimation is an attractive approach with notable approaches utilizing black box optimization schemes such as Finite Differences [Chen et al., 2017], Natural Evolutionary Strategies [Ilyas et al., 2018, Jiang et al., 2019], learned priors in a bandit optimization framework [Ilyas et al., 2019], meta-learning attack patterns [Du et al., 2020], and query efficient.

Defenses. In order to protect against the security risk posed by adversarial examples there have been many proposed defense strategies. Here we provide a non-exhaustive list of such methods. Broadly speaking, most defense approaches can be categorized into either robust optimization techniques, gradient obfuscation methods, or adversarial example detection algorithms [Xu et al., 2019]. Robust optimization techniques aim to improve the robustness of a classifier by learning model parameters by incorporating adversarial examples from a given attack into the training process [Madry et al., 2018, Tramèr et al., 2018, Ding et al., 2020]. On the other hand obfuscation methods rely on masking the input gradient needed by an attacker to construct adversarial examples [Song et al., 2018, Buckman et al., 2018, Guo et al., 2018, Dhillon et al., 2018].

In adversarial example detection schemes the defender seeks to sanitize the inputs to the target model by rejecting any it deems adversarial. Often this involves training auxiliary classifiers or differences in statistics between adversarial examples and clean data [Grosse et al., 2017, Metzen et al., 2017, Gong et al., 2017].