Deep learning with Lipschitz constraints

Louis Béthune

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Université de Toulouse

Doctorat de l'Université de Toulouse

préparé à l'Université Toulouse III - Paul Sabatier

Apprentissage profond avec contraintes Lipschitz

Thèse présentée et soutenue, le 7 février 2024 par

Louis BÉTHUNE

École doctorale
EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse

Spécialité
Informatique et Télécommunications

Unité de recherche
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Mathieu SERRURIER

Composition du jury
M. Loic BARTHE, Président, Université Toulouse III - Paul Sabatier
M. Gabriel PEYRE, Rapporteur, CNRS Paris-Centre
M. Alexandre ALLAUZEN, Rapporteur, Université PSL
M. Remi FLAMARY, Examinateur, Institut Polytechnique de Paris
Mme Louise TRAVE-MASSUYES, Examinatrice, CNRS Occitanie-Ouest
M. Mathieu SERRURIER, Directeur de thèse, Université Toulouse III - Paul Sabatier

Membres invités
M. Jean-Michel Loubes, Universite Toulouse III - Paul Sabatier
M. Andrés Troya-Galvis, Thales Alenia Space

Deep Learning with Lipschitz Constraints

Louis Béthune

2023

Abstract

This thesis explores Lipschitz constraints with respect to the input in deep learning. They have been used in the past for computational optimal transport and certifiability against adversarial attacks. However, some of their properties were still unknown; in particular, it was not clear if arbitrary classification tasks could be solved by these networks.

In this thesis, I answer positively this question by showing there exists an intrinsic accuracy/robustness tradeoff for Lipschitz neural networks, controlled by the entropic regularization of the loss. Furthermore, they benefit from various generalization guarantees, including ones that are architecture-independent.

Then, the thesis shows that Signed Distance Function (SDF) estimation can be reformulated as a regularized optimal transport problem, that can be solved with Lipschitz networks trained in an adversarial setting. The methods can be used for robust One Class learning or more stable implicit surface parametrization.

Finally, I show that Lipschitzness with respect to inputs is tightly linked to robustness with respect to parameters. This yields a "backpropagation for bounds" algorithm, that can automatically compute loss gradient bounds for Lipschitz networks. In particular, this allows for training with robustness and differential privacy guarantees.

Remerciements

Je tiens tout d'abord à remercier les membres de mon jury, Mr Allauzen, Mr Peyré, Mme Louise Travé-Massuyès, Mr Barthe and Mr Flamary. En particulier, les travaux de Mr Allauzen ont été une source d'inspiration importante en lien direct avec mon sujet de thèse. Je dois aussi témoigner mon vif intérêt pour les travaux de Mr Flamary et Mr Peyré en transport optimal. Merci à Mr Allauzen et Mr Peyré pour leur méticuleux travail de rapporteur, vos remarques ont été très utiles à la structuration de la soutenance.

Je tiens aussi à exprimer ma gratitude à mon directeur de thèse Mathieu Serrurier qui m'a accompagné, encouragé et mentoré. Ta bienveillance et ton écoute m'ont porté ces trois dernières années. Je tiens aussi à remercier le directeur de ma chaire ANITI Jean-Michel Loubes pour son combat administratif constant. Merci à Andres Troya-Galvis pour l'intérêt continu que tu as porté à mon travail; j'aurai aimé pouvoir garder une proximité plus grande avec les sujets de Thalès mais la sérendipité m'a mené ailleurs.

Merci à Mathieu Blondel, pour m'avoir fait confiance en me prenant en stage chez Google Brain, pour m'avoir initié à la contribution aux projets OpenSource et m'avoir fait découvrir le monde merveilleux de la différentiation implicite.

Je tiens aussi à remercier le programme ANITI qui a financé ma thèse, ainsi que tout son personnel, qui s'est battu pour s'assurer que les thèses soient menées dans de bonnes conditions. Merci à Aurélien Garivier d'avoir cru en moi et de m'avoir encouragé à faire une thèse à Toulouse, ce fut un excellent conseil.

Merci au projet DEEL pour son soutien and sa bonne ambiance, vous avez joué un rôle important dans mon épanouissement professionnel. Franck pour ta gentillesse et ton engagement, tu es un modèle pour moi. Sébastien, qui a sauvé un de mes théorèmes en découvrant une erreur pernicieuse.

Thomas, j'ai été très heureux de te prendre en stage, j'ai beaucoup appris à tes côtés et j'ai hâte de voir ce que tu feras dans la suite de ta carrière.

Le gang des cool kids et le plateau Confiance, Corentin, Lucas, Louise, Antonin, Fred, Mikaël, Estèle, Fanny, Léo, David B, Etienne, Mélanie, vous avez égayé mes journées et surtout mes soirées. Un merci spécial à David V et Luca qui m'ont permis de manger à l'ENAC pour moins cher, parfois malgré eux. Victor tu es ma boussole éthique en science et je te souhaite le meilleur dans tes projets. Alberto, toi qui m'as beaucoup appris.

Paul, Yannick, Joseba pour nos moments de partage au Bikini et ailleurs.

Thibaut, Agustin, Thomas, je porte dans mon coeur notre voyage en Argentine et les nombreuses discussions scientifiques qui l'ont accompagné.

Joël, Danaé, Nicolas, Guillaume[1] pour notre aventure à la coloc-scopie. Mes autres compères Lyonnais Alexandre, Léo, Elena, Mathieu.

Erwan, Héloïse, Koma, Elisa, Rémi, pour notre amitié durable.

Mes parents pour votre support constant. Mon petit frère Henri, ma petite soeur Clothilde, j'aime les adultes que vous êtes devenus et c'est un plaisir de vous voir grandir.

Pour finir, Marie qui fut à mes côtés ces trois dernières années, et qui a vu ce projet de ses balbutiements jusqu'à sa finalisation. Pour ta fidélité, ton écoute, ta compréhension, et ton soutien inébranlable, merci.

[1] Cet article que nous avons écrit ensemble n'est que la suite logique d'une collaboration scientifique démarrée en 2015.

Contents

Chapter 1

Introduction

1.1 Deep learning and open challenges

In this chapter, we attempt to define deep learning as a field and to identify the main challenges it faces. Then, we give the first theoretical tools that are widely used throughout the thesis. After, we motivate the study of Lipschitz constraints in deep learning by performing a literature review of their usage. Finally, we summarize the main contributions of this work.

Contents

For simplicity of the exposure, we start by defining feedforward neural networks. The affine transformation takes the form $h \mapsto f_t(W_t, h) + b_t$, where b_t is the **the bias**, and W_t the "linear" weights whose exact form depends on the layer (densely connected pattern, convolution, channel/token mixing, etc).

Definition 1 (Feedforward neural network)**.** *A feedforward neural network of depth T, with input space $\mathcal{X} \subset \mathbb{R}^n$, and with parameter space $\Theta \subset \mathbb{R}^p$, is a parameterized function $f : \Theta \times \mathcal{X} \to \mathcal{Y}$ defined by the following recursion:*

$$\begin{aligned} h_0(x) &:= x, & z_t(x) &:= f_t(W_t, h_{t-1}(x)) + b_t, \\ h_t(x) &:= \sigma(z_t(x)), & f(\theta, x) &:= z_{T+1}(x). \end{aligned} \tag{1.1}$$

The set of parameters is denoted as $\theta = (W_t, b_t)_{1 \leq t \leq T+1}$, the output space as $\mathcal{Y} \subset \mathbb{R}^K$ (e.g logits), the layer-wise non-linearity as $\sigma : \mathbb{R}^n \to \mathbb{R}^n$. Here, f_t denotes a bilinear function.

In the following, AllNet will denote these networks.

Remark 1.1. The above formulation is extremely general.

The non-linearity σ was historically the sigmoid $x \mapsto \frac{1}{1+\exp -x}$ or the hyperbolic tangent function $x \mapsto \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Now, practitioners typically rely on ReLU $x \mapsto \max(0, x)$ and its variants: LeakyReLU, SiLU, GeLU, SELU, Softplus, etc. All these modern activations share the property of being close to zero on one side of the graph (when $x \to -\infty$), and almost linear on the other (when $x \to +\infty$). In this sense, they can all be interpreted as the integral of some smooth relaxation of the step function $s(x) = \mathbb{1}_{x \geq 0}$, that first appeared in the perceptron algorithm (Rosenblatt, 1957).
The bi-linear operator f_t can denote a matrix-vector product, a convolution, any concatenation or splitting method, a mean reduction, a residual connection, a batch normalization Ioffe and Szegedy (2015) in *test mode* (with fixed statistics).
Furthermore, other operations like layer normalization Ba et al. (2016) or group normalization Wu and He (2018) can be modelized as a non-linearity. Even the "self-attention" mechanism (Vaswani et al., 2017) can be modelized by stacking linear layers (with some sparsity structure) and a *softmax* non-linearity.

The most common architectures, such as Convolutional Neural Networks (CNNs), Fully Connected Networks (FCNs), Residual Networks (ResNets), patch-based classifiers (like MLP-Mixers), and Transformers, all fall under the category of feed-forward networks.

Remark 1.2. Universal Approximation theorem(s).

Under extremely mild assumptions on the non-linearity σ, AllNet networks benefit from universal approximation theorem in $C(\mathcal{X}, \mathbb{R}^K)$, a classical result of literature (Cybenko, 1989; Hornik, 1991). See also Hassoun et al. (1995); Pinkus (1999); Haykin (1998) for a survey.

More than a single result, it is actually of family of results that depends on additional assumptions such as the exact form of σ. Some variants are easier than others, for example when the activation has finite limits $\sigma(-\infty) < \sigma(+\infty)$. The main limitation is that σ cannot be a polynomial, in particular, it cannot be linear, otherwise, the whole network would degenerate to an over-parametrized linear function.

Warning 1.1. Beware of false sirens.

The universal approximation theorem often serves as a justification for the widespread use of neural networks[a]. In the language of learning, this is a class of functions *without* bias. This contrasts with most of the classical literature on machine learning, which traditionally focused on models with some sort of bias to mitigate variance, following findings of the seminal work of Valiant (1984).
Moreover, this is not the only parametrized class of functions benefiting from universal approximation theorems: polynomials with Stone–Weierstrass theorem (Stone, 1937), Fourier basis (Trigub and Belinsky, 2004), and wavelets basis (Kaiser and Hudgins, 1994); they all share this surprisingly common property. Therefore, the expressiveness of neural networks is not enough to explain their empirical success.
However, neural networks seem to have an asymptotic advantage regarding the number of

parameters required to achieve error at most ε, see the works of Cheridito et al. (2021); DeVore et al. (2021); Yarotsky (2017).

[a]This can be found as far as on the X (ex-Twitter) threads of some "data-science influencers".

In recent years research has focused a lot on the *implicit bias* induced by the optimization procedure itself. Instead of looking for the minimizers of the empirical risk, the role of the optimizer is now studied as well. It revealed that the optimizer itself induced a bias on the solutions returned. This interlacing between the hypothesis class on one hand, and the optimization procedure on the other hand, is now believed to play a central part in explaining the generalization of neural networks.

The difference between the two paradigms is detailed below:

$$\underbrace{\theta^* \in \arg\min_{\theta} \mathbb{E}_{z\sim\mathcal{D}}[\mathcal{L}_\theta(z)]}_{\text{Empirical Risk Minimization}} \quad \neq \quad \underbrace{\theta^T \sim \mathcal{A}\left(\theta \mapsto \nabla_\theta \mathbb{E}_{z\sim\mathcal{D}}[\mathcal{L}_\theta(z)]\right)}_{\text{Implicit Regularization of Stochastic Optimization}} \tag{1.2}$$

where:

- $\mathcal{D}$ the dataset: a finite sample of size n.
- $\mathcal{L}_\theta(z)$ a loss parametrized by θ. For example, in supervised learning task with input x and target y, the loss takes the form $\mathcal{L}(f_\theta(x), y)$ with f_θ a neural network.
- $\mathcal{A}$ a *stochastic* algorithm that attempts to find the optimum, in a stochastic manner, using stochastic gradients $\theta \mapsto \nabla_\theta \mathbb{E}_{z\sim\mathcal{D}}[\mathcal{L}_\theta]$ evaluations, and returning some θ^T after a finite time T.

The optimization landscape of neural networks is notoriously not convex, often non-smooth (e.g because of ReLU), and stochastic because of the mini-batchs, therefore it is expected that an algorithm $\mathcal{A}$ running for a finite amount of time to fail to return the true global optimum. But even though the algorithm could find a global optimum, there are no reasons for it to be unique. Indeed, for a finite dataset $\mathcal{D}$, without regularization on model's weights, the predictions of the model on $\mathcal{D}$ are determined by $\arg\min$, but outside of it, on the whole domain $\mathbb{R}^m$, where there are typically multiple behaviors possible. Some of the empirical minimizers will generalize, and some others, not so much. So, even if $\mathcal{A}$ could find a global minimum (and there are many of these), it needs to find one of the "good ones", i.e. one that generalizes well.

Therefore, the role of the optimizer in learning has provoked an important number of works, among which the study of the "double-descent" phenomenon (Zhang et al., 2021b; Belkin et al., 2019; d'Ascoli et al., 2020; Nakkiran et al., 2021). For example see Schaeffer et al. (2023) and references therein.

The definition of neural network is ever-changing as new architectures are created over the years. The frontier between what is considered as preprocessing, what is considered a layer, and what is considered as part of the loss becomes more and more blurry with the appearance of "optimization layers" (see chapter 3 for a short introduction), or with the new data augmentations like *mixup* (Zhang et al., 2017). While the prototype of the optimizer is "vanilla" Stochastic Gradient Descent, now it is frequent to incorporate momentums (with and without Nesterov (Nesterov, 1983) correction), gradient rescalings, gradient clipping, adaptive learning rates, etc. which makes the characterization of the optimizer extremely tedious.

It is common for theorists to create precise concepts (like in definition 1) because it allows reasoning, but it may also fail to faithfully capture the dynamics at play. For example, most theorists will assume that computations are done on $\mathbb{R}$, whereas it is done in the field $\mathcal{F}$ of representable numbers in floating arithmetic. The most frequent precision used is *float32* due to typical GPU requirements, but recent interest in frugality or training cost sparked interest for *float16*, and even

sometimes lower precision. While numerical inaccuracies seem like a trivial matter, they play an important role as a regularizer, as explored in chapter 4.

Takeaways

It is more useful to think about Deep learning as a *paradigm* in which to think about problems and solutions, than a specific technology. The details are ever-changing at time of writing, but in modern deep learning the following principles remain:

1. Parametric approximation of functions, typically organized into layers, but typically not convex nor smooth. The architecture of the approximator captures some (partial) invariance of the data, like invariance under translation for convolutional layers for example.
2. The function approximator is locally linear in its parameters, hence subgradients exist. The gradient is computed with Autodiff automatically.
3. The optimization is *stochastic*: only a subset of samples is used at each step.
4. Capacity to handle huge amounts of data in high dimensions, including with label noise or corrupted data. Capacity to handle data augmentations. This implies to rely on algorithms whose runtime is linear in the size of the train set.
5. Can leverage parallelism, especially the one offered by GPU, in low-precision arithmetic (like matrix-vector products).

Each of those items brings its implicit regularization through randomness, constraints, optimization landscape, etc. Each of those five contributes to the generalization capabilities and efficiency of the method on real-world tasks. Every algorithm that falls within those principles may be argued to belong to the "deep learning" paradigm. In particular, this does *required* to use neural networks: ODE (Chen et al., 2018), Fixed point computations (El Ghaoui et al., 2021), or other arbitrary computations graphs (Weber et al., 2019). This view is summarized in Figure 1.1.

The thesis will focus on **1. Architectural constraints**, and **2. Autodifferentiation**. More specifically, we will study how to incorporate Lipschitz constraint in Deep learning pipeline in chapters 2,4,5,6 and convexity constraints in chapter 7. Implementing these constraints require looking at the computation graph and ensuring that every constraint is compatible with automatic differentiation. This is studied in chapters 3,7 and 6.

1.2 Notations and tools

Most of the tools of the thesis will be introduced on the fly in the chapters that need it. However, some of these are relevant to the whole thesis and deserve to be introduced here.

Definition 2 (Lipschitz constant)**.** *The function $f : \mathbb{R}^m \to \mathbb{R}^n$ is said ℓ-Lipschitz for l_2 norm if for every $x, y \in \mathbb{R}^m$ we have:*

$$\|f(x) - f(y)\|_2 \leq \ell \|x - y\|_2. \tag{1.3}$$

Per Rademacher's theorem (Simon et al., 1983), its gradient is bounded: $\|\nabla f\| \leq \ell$. Reciprocally, continuous functions gradient bounded by ℓ are ℓ-Lipschitz.

Another tool that will come handy is related to the *convexity* of a function.

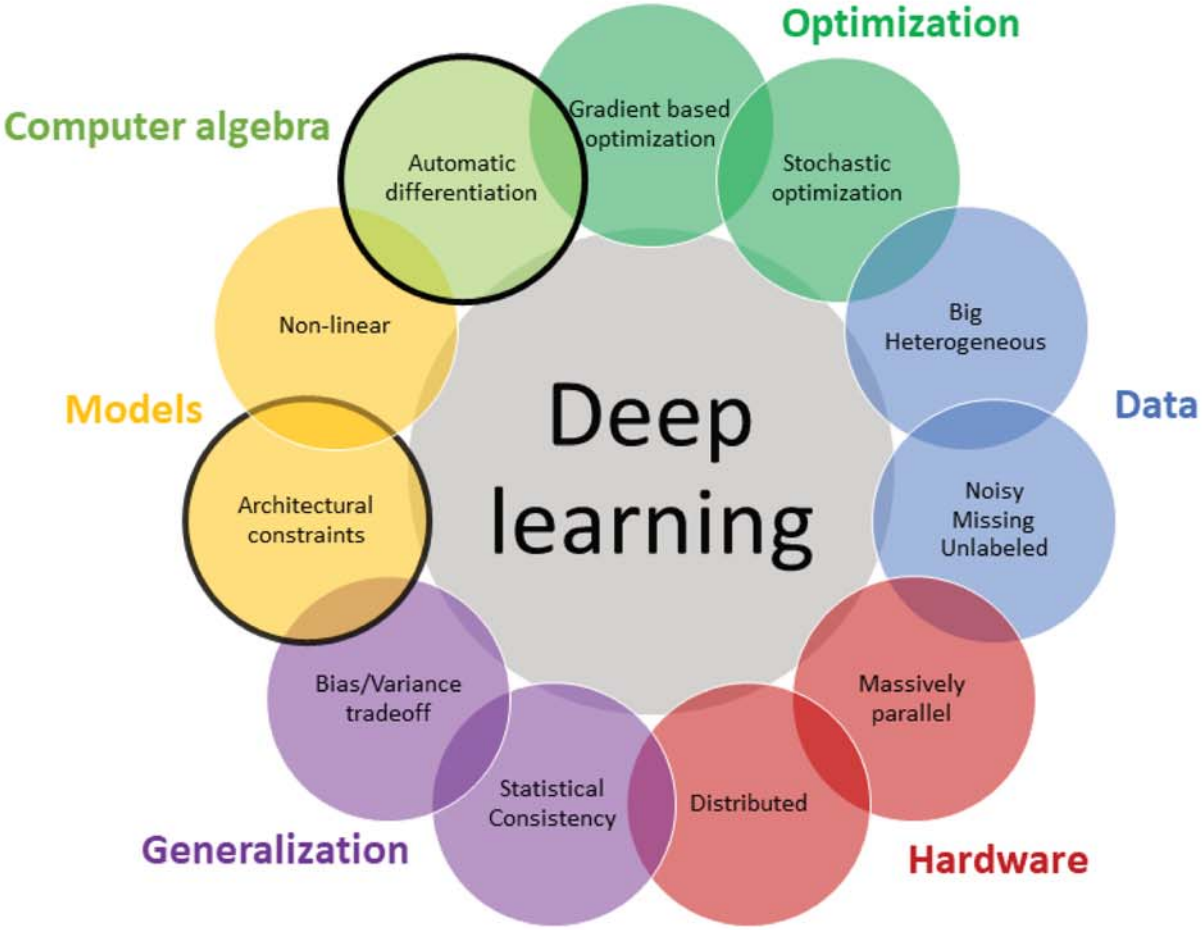


Figure 1.1: **Deep learning is a paradigm that covers many areas**. Black circles highlight the main topics to which this thesis contributes.

Definition 3 (Convex function)**.** *The function $f : \mathbb{R}^m \to \mathbb{R}$ is said to be convex if and only if for all $x, y \in \mathbb{R}^m$, for all $\lambda \in [0, 1]$, the following inequality holds:*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y). \tag{1.4}$$

Furthermore, if f is differentiable, then its gradient $\nabla_x f$ is cyclically monotone (Peyré et al., 2017). Finally, if f is twice-differentiable its Hessian $\mathcal{H}_x f$ is a positive semidefinite matrix.

In the following, the notation $\|\cdot\|_2$ will denote the common Euclidean norm for vectors, i.e.:

$$\|x\|_2 = \sqrt{\sum_i x_i^2}, \tag{1.5}$$

whereas for linear operators $x \mapsto Wx$ it will denote the spectral norm:

$$\|W\|_2 = \max_{\|x\|_2 \leq 1} \|Wx\|_2 = \sigma_{\max}, \tag{1.6}$$

where $\sigma_{\max}$ is the largest singular value in the SDV of W.

Warning 1.2. Vector or matrix?

The difference between vectors and operators is *not* tied up to the "shape" of the object, but rather dependant on the context. For example, a black-and-white image is a 2D matrix but it should be considered a vector. Linear operators acting on RGB images are *technically* 6D tensors, but it is more useful to think about them as 2D matrices acting on flattened images.

1.2.1 Optimal transport

Some of the results of this thesis are tightly linked to optimal transport. We give below the definition of Wasserstein-p distance, as found in Villani (2008) (Definition 6.1).

Definition 4 (Wasserstein-p distance)**.** *Let $d : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ be a metric. For any two measures P and Q on $\mathbb{R}^m$ the Wasserstein-1 distance is defined by the following optimization problem:*

$$\mathcal{W}_p^p(P, Q) := \inf_{\pi \in \Pi(P,Q)} \int_{\mathbb{R}^m} (d(x, y))^p \mathrm{d}\pi(x, y) \tag{1.7}$$

where $\Pi(P, Q)$ denote the set of measures on $\mathbb{R}^m \times \mathbb{R}^m$ whose marginals are P and Q respectively. Equivalently we can write:

$$\mathcal{W}_1(P, Q) := \inf_{\substack{Law(X)=P \\ Law(Y)=Q}} \mathbb{E}[d(X, Y)]. \tag{1.8}$$

The right hand side of 1.7 corresponds to a Kantorovich problem. It is often presented as a relaxation of the Monge problem:

$$\inf_{T\#P=Q} \int_{\mathbb{R}^m} d(x, T(x))\mathrm{d}p(x). \tag{1.9}$$

In this context, the (optimal) $T : \mathbb{R}^m \to \mathbb{R}^m$ is called the (optimal) *Monge map*. For arbitrary measures, the two problems are different. The Kantorovich problem always admits a solution. This is not the case with the Monge problem. For example, for discrete measures with supports of size

$|P| < |Q|$, it is impossible to fulfill the pushforward condition $T\#P = Q$ since $|T\#P| \leq |P| \leq |Q|$. Nonetheless, when both P and Q admit a density (w.r.t Lebesgue measure) then the two problems happen to be equivalent. In this case, the Monge problem corresponds to a deterministic coupling $\pi(x, \cdot) = \delta_{T(x)}$.

By Remark 6.3 of Villani (2008) the Wasserstein-1 distance is the Kantorovich-Rubinstein distance:

$$\mathcal{W}_1(P, Q) = \sup_{f \in \text{Lip}_1(\mathcal{X}, \mathbb{R})} \mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{z \sim Q}[f(z)]. \tag{1.10}$$

In our case we are working with neural networks that are Lipschitz w.r.t. $l2$ distance, so we have $d(x, y) := \|x - y\|_2$. This chapter is one of the first motivations for the design of Lipschitz networks, and a strong inspiration for most of the work throughout the thesis, notably chapters 4 5 6.

Dual formulations of optimal transport

We can also mention a useful duality result related to Wasserstein-2 distance (see Peyré et al. (2017) or Korotin et al. (2021) for example):

$$\mathcal{W}_2^2(P, Q) = \sup_{f \in \text{Convex}(\mathcal{X}, \mathbb{R})} \mathbb{E}_{x \sim P}[f(x)] + \mathbb{E}_{z \sim Q}[f^c(z)], \tag{1.11}$$

where f^c is the c-transform defined as:

$$f^c(x) := \min_y c(x, y) - f(y) \tag{1.12}$$

for $c(x, y) = \|x - y\|$. This result will not be used *as-is*, but it shows the connections between optimization over convex functions on one side and optimal transport on the other side. This is explored further in chapter 7.

1.2.2 Losses

We detail below the losses (and activations) that will be used in this work.

The Binary Cross-Entropy (BCE) loss (also called log loss) is among the most popular choices of loss within the deep learning community. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a neural network. For an example $x \in \mathbb{R}^n$ with label $y \in \mathcal{Y}$, and $\sigma(x) = \frac{1}{1+\exp(-x)}$ the logistic function mapping logits to probabilities, the BCE is written $\mathcal{L}_\tau^{bce}(f(x), y) = -\log \sigma(y\tau f(x))$, with (inverse) temperature [1] scaling parameter $\tau > 0$. This hyper-parameter of the loss defaults to $\tau = 1$ in most frameworks such as Tensorflow or Pytorch. Note that $\mathcal{L}_\tau^{bce}(f(x), y) = \mathcal{L}_1^{bce}(\tau f(x), y)$ so we can equivalently tune τ or the Lipschitz constant L. We show in Chapter 4 that **for LipNet1 the temperature τ allow to control the generalization gap**.

The Hinge loss $\mathcal{L}_m^H(f(x), y) = \max(0, m - yf(x))$ with margin $m > 0$ is also of particular interest.

[1] The name "temperature" finds its roots in statistical physics, notably a link with Shannon entropy, which is itself linked with the conventional thermodynamics's entropy, as we will see later.

1.3 Motivations and open problems

In this thesis, we focus mainly on Lipschitz constraints. We also briefly explore the convexity constraints in chapter 7. The motivation behind constraining and studying the Lipschitz constant of neural networks is two-fold. First, we are looking to strive for a better understanding of learning dynamics in neural networks. Secondly, we are interested in learning with guarantees: robustness, privacy, or fairness. The societal implications of these topics are largely discussed in surveys such as Akhtar and Mian (2018), Liu et al. (2020) and Mehrabi et al. (2021). Their content will not be detailed here.

A literature review on Lipschitz constraints in neural networks reveals their usage can be roughly sorted under four main categories:

1. **Robustness against adversarial attack.** Historically one of the first motivations for their study. The link between robustness against adversarial attacks and Lipschitz constant is as old as the field of adversarial robustness itself (Szegedy et al., 2014).
2. **Lipschitz constraints for optimal transport.** The link between Lipschitz constraints and optimal transport is straightforward through the Kantorovich-Rubinstein duality and was the origin of Wasserstein Adversarial Neural Networks Arjovsky et al. (2017)[2]
3. **To obtain generalization guarantees.** In this context the Lipschitz hypothesis appears as a vessel for different quantities related to the model's complexity, which frequently appears in generalization results. The Lipschitz constant of hypothesis class frequently appears in theorems, either directly or through related complexity measures.
4. **Stability of the training.** In various applications, the Lipschitz (or sometimes the even stronger *orthogonality* condition) appears as a natural solution to overcome instabilities in training, divergence, or other unwanted behaviors.

All those topics are (distantly) related:

- $3 \leftrightarrow 4$: stabilizing the training procedure yields algorithms that are more algorithmically stable Bousquet and Elisseeff (2002), which in known to yield generalization guarantees.
- $1 \leftrightarrow 3, 4$: the instabilities that arise during the training of neural networks typically come for their high expressiveness. On some ill-formed losses, divergence of the training procedure can be observed. Robustness against adversarial attacks is arguably a form of regularization that may mitigate these phenomena.
- $1 \leftrightarrow 2$: a first attempt at drawing a link between those two field is made in chapter 4.

1.3.1 Certifiable robustness

LipNet1 networks provide robustness radius certificates against adversarial attacks (Tsuzuku et al., 2018). There is no need for adversarial training (Madry et al., 2018) that fails to produce guarantees, or for randomized smoothing (Cohen et al., 2019) which is costly.

Confusingly, any network of AllNet has a finite Lipschitz constant (since weights are bounded), but computing it is NP-hard (Scaman and Virmaux, 2018). Only a loose upper bound can be cheaply estimated: $\text{Lip}(f) \leq \text{Lip}(\sigma)^d \Pi_{i=1}^d \|W_i\|_2$ using the property that $\text{Lip}(f_d \circ f_{d-1} \circ \ldots \circ f_1) \leq \Pi_{i=1}^d \text{Lip}(f_i)$. In practice, this bound is often too high to provide meaningful certificates and besides, AllNet networks are known to have a very small robustness radius (Szegedy et al., 2014).

Definition 5 (Adversarial Attack). *For any classifier $c : \mathcal{X} \to \mathcal{Y}$, any $x \in \mathbb{R}^n$, consider the following*

[2] Interestingly, Goodfellow was looking for defense against adversarial attacks when he discovered the first GAN Goodfellow et al. (2014), a remarkable example of serendipity.

optimization problem:

$$\epsilon = \inf_{\delta \in \mathbb{R}^n} \|\delta\| \text{ such that } c(x+\delta) \neq c(x). \tag{1.13}$$

δ is an adversarial attack, $x+\delta$ is an adversarial example, and ϵ is the robustness radius of c in x. The smallest $\|\delta\|$ achievable by $x \in \mathcal{X}$ is the ***minimum robustness radius*** *of c.*

Lipschitz constraints yield robustness certificates against adversarial attacks.

Property 1 (Local Robustness Certificates (Tsuzuku et al., 2018))**.** *For any $f \in Lip_L(\mathcal{X}, \mathbb{R})$ the robustness radius ϵ of binary classifier $sign \circ f$ at example x verifies $L\epsilon \geq |f(x)|$.*

Property 1 can be extended to the multiclass case, with the use of multiclass margins.

Property 2 (Multiclass Local Robustness Certificates)**.** *For any $f \in Lip_L(\mathcal{X}, \mathbb{R}^K)$ the robustness radius ϵ of classifier $\hat{k} := \arg\max_k f_k(x)$ verifies $\sqrt{2}L\epsilon \geq (f_{\hat{k}}(x) - \arg\max_{i \neq \hat{k}} f_i(x))$.*

Note that these definitions of adversarial attack **do not** take into account the true label: it focuses on the robustness of the decision, irrespective of the ground truth label. For practical applications, however, it is common to consider that the robustness radius of a misclassified example is zero: this way, the certified accuracy never exceeds the clean accuracy.

Computing these certificates is straightforward and does not increase runtime, contrary to methods based on bounding boxes or abstract interpretation (see section 2.1). Controlling the Lipschitz may even benefit robustness certificates based on randomized smoothing (Delattre et al., 2023a).

In Hu et al. (2023a) the authors propose a comprehensive framework to design ℓ-Lipschitz networks, leveraging both real and generated data to improve the generalization and the certificates. In Hu et al. (2023b) let the Lipschitz constant of individual layers evolve freely, but they rely on the product bound to produce certificates. The case of certification against l_∞-attacks is handled by Zhang et al. (2021a) and Zhang et al. (2022). The link between Lipschitz constant, dimension of parameter space and robustness is estavblished in Bubeck and Sellke (2021). Finally, the recent work of Mangal et al. (2023) and Leino (2023) explores the limits of Lipschitz-based certificates.

1.3.2 Optimal transport and generative models

Lipschitz constraints arise in the context of optimal transport (Arjovsky et al., 2017). The optimal transportation plan G is used as a generative model by minimizing $\mathcal{W}_1(G\#P, Q)$. In Gulrajani et al. (2017), authors show that the potential f of the Kantorovich-Rubinstein dual transport problem verifies $\|\nabla_x f(x)\| = 1$ almost everywhere on the support of the distributions $\mathbb{P}_{XY}$. In Tanielian and Biau (2021) the benefit of GroupSort for WGAN training is studied. The convergence rate of WGAN discriminators for Wasserstein-distance approximation is also studied in Gao et al. (2023). We illustrate an optimal transportation plan between two moons for Euclidean cost in Figure 1.2, based on 1-Lipschitz neural networks. Optimal transport itself can be used for classification (Serrurier et al., 2021) or XAI (Serrurier et al., 2023). In this context, 1-Lipschitz neural networks trained with the appropriate loss are shown to produce more interpretable gradients than conventional networks (see figure 1.3). More recently, high-quality embeddings from DreamSim (Fu et al., 2023) have been distilled into 1-Lipschitz backbones (Ghazanfari et al., 2023) to produce robust perceptual similarity metrics.

Among possible applications of optimal transport based on Lipschitz networks, we can mention cell counting Ding et al. (2023). We can also mention Coiffier et al. (2020) for the applications of WGAN in 3D modeling of geological layers.

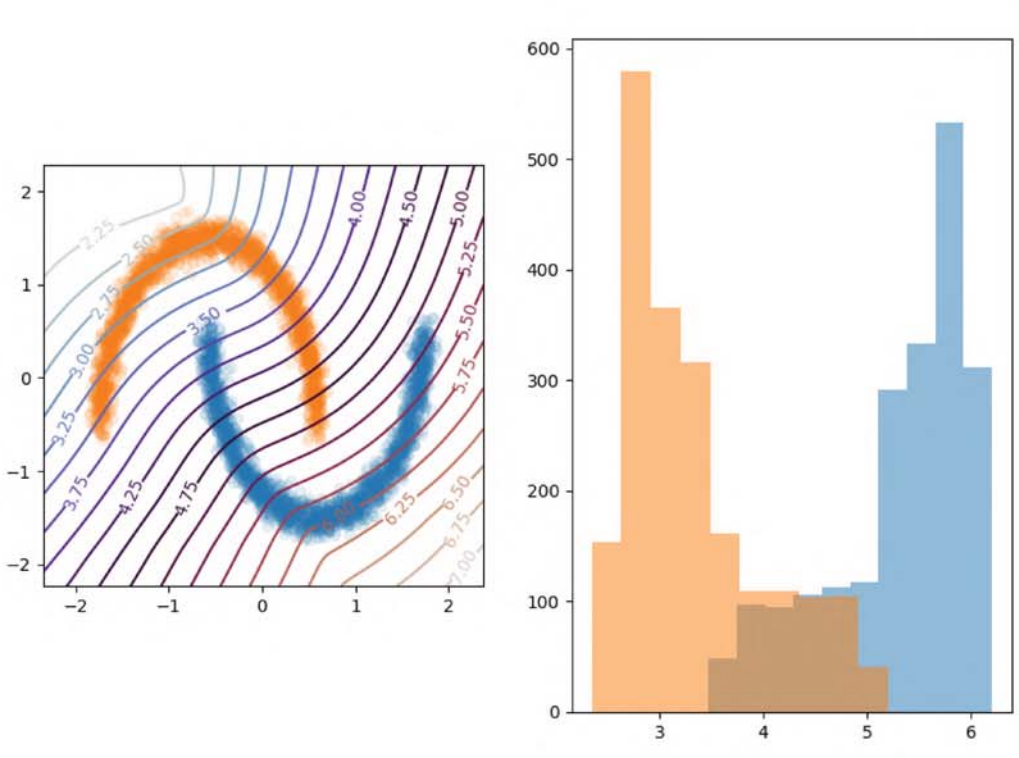


Figure 1.2: **Level sets of the 1-Lipschitz potential** f (left), and **histogram of scores** (right) when computing the Monge map between the two moons, for euclidean cost $c(x, y) = \|x - y\|_2$. This relies on 2 layers neural networks with orthogonal matrices (using Björck projection) and Groupsort activation function.

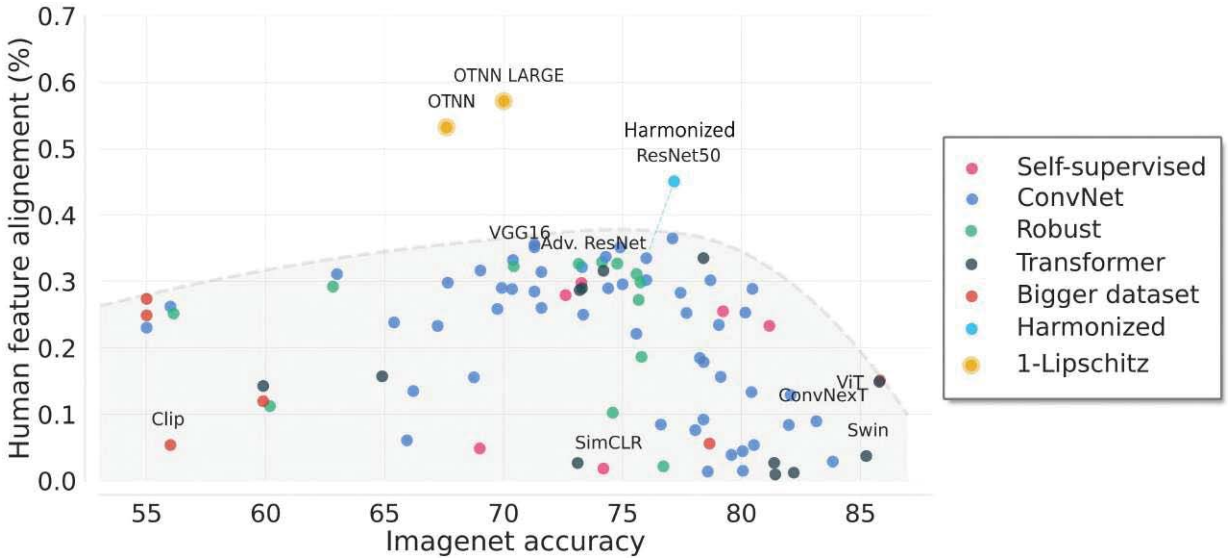


Figure 1.3: **1-Lipschitz neural networks trained with optimal transport losses are aligned with Human attention (Serrurier et al., 2023)**. This figure is a courtesy of Mathieu Serrurier. The study shows that the Saliency Map of the 1-Lipschitz neural network (named OTNN here) is highly aligned with human attention on the ClickMe dataset (Linsley et al., 2017).

1.3.3 Generalization guarantees

In the seminal work of von Luxburg and Bousquet (2004) a link is established between Lipschitz classifiers and linear large-margin classifiers. Generalization bounds for a large class of Lipschitz classifiers are provided by the work of Gottlieb et al. (2014) using Vapnik–Chervonenkis theory. Other generalization bounds related to spectral normalization can be found in Bartlett et al. (2017). Links between adversarial robustness, large margins classifiers and optimization bias are studied in Faghri et al. (2021); Finlay et al. (2018); Jiang et al. (2019). The importance of the loss in adversarial robustness is studied in Pang et al. (2019). In Tsuzuku et al. (2018), the control of Lipschitz constant and margins are used to guarantee robustness against attacks. A link between classification and optimal transport is established in Serrurier et al. (2021) by considering a hinge regularized version of the Kantorovich-Rubinstein dual objective. Note that the PAC-Bayes theory (Shawe-Taylor and Williamson, 1997) can also be applied to the question of adversarial robustness (Viallard et al., 2021). The Dirichlet energy is a distantly (but related) measure of complexity, that shares similarities with the Lipschitz constant, and was shown to correlate with better generalization (Dherin et al., 2022).

1.3.4 Lipschitz and orthogonality constraints for stability

In Zhang et al. (2018b) the use of Householder reflectors is discussed to solve the vanishing gradient issues that arise in the training of Recurrent Neural Networks. Orthogonal kernels are also of special interest in the context of normalizing flows (Hasenclever et al., 2017). Orthogonalization methods have been used on the output of *base* classifiers (Mashhadi et al., 2021) for ensemble methods. Lipschitz constraints are used in reinforcement learning to stabilize the policy, notably on continuous control tasks (Song et al., 2023), or to avoid catastrophic divergence of the optimizer (Gogianu et al., 2021). In the context of graph neural networks, Dasoulas et al. (2021) proposed to remove the exploding gradient phenomenon arising in the graph attention layer, using matrix norm constraints. They have also been used for fairness purposes (Jia and Zhang, 2023). Finally, we can mention

all the work in Gupta (2023) and its companion articles such as Gupta et al. (2022a,b), where the Lipschitz constant of a network is optimized and adjusted online. Lipschitz constraint have also shown promises for continual learning in the work of Bonicelli et al. (2022).

1.4 Contributions of this thesis

This thesis focuses largely on two topics: architectural constraints in deep learning, and more specifically Lipschitz constraints, and the use of back-propagation to train these constrained architectures. The short-term objective of studying Lipschitz constraints in learning amounts to their traditional applications such as robustness certification against adversarial attacks, or differentially private learning. But another longer-term is to strive better understanding of deep learning in general, and Lipschitz constraints appear as a natural lens to do so.

1.4.1 Research contributions

The contributions of the thesis are twofold: one part is dedicated to Lipschitz constraints in learning with some applications, while the second part explores the possibilities offered by the paradigm of differentiable programming.

The main part of the manuscript focuses on Lipschitz constraints and their applications.

Chapter 2 is a brief literature review on the topic of Lipschitz networks, covering their parametrizations. It also covers the topic of "orthogonal layers", which are not only 1-Lipschitz but also fulfill the Eikonal equation $\|\nabla_x f(x)\| = 1$ almost everywhere.

Chapter 3 showcase different applications of the implicit function theorem to compute derivatives of optimization problems in various contexts, like Non-Negative Matrix Factorization, or the projection onto the Stiefel manifold for the parametrization of orthogonal layers.

Chapter 4 asks the question of the expressiveness of the 1-Lipschitz function class for classification purposes. It derives the theoretical guarantees that Lipschitz constraints can bring to classification tasks, including robustness, and generalization. An opening to calibration in the presence of a distribution shift is discussed.

Chapter 5 explores the application of these constraints to the parametrization of signed distance functions, which allows applications in robust one-class learning, anomaly detection, and implicit surface parametrization. The links with Energy Based Models (EBM) are discussed.

Chapter 6 proposes to use Lipschitz networks in the context of differential privacy: it is based on a variation of the backpropagation algorithm, to automatically derive bounds.

Chapter 7 presents an attempt at using Lipschitz and convexity constraints for Wasserstein-2 distance estimation, with applications to counterfactual fairness or multivariate quantiles with the center outward map. Finally, a fully neural multivariate quantile regression algorithm is discussed.

Finally, the conclusion in chapter 8 is intended as a research statement of possible future work, and broadens the topic of Lipschitz networks by drawing links with existing tools.

Appendix C shows how automatic differentiation can be used to optimize the hyper-parameters of a Gaussian process on distribution, by backpropagating through a GP likelihood and Sinkhorn algorithm.

1.4.2 Software contributions

During an internship in Google Brain, I contributed to `jaxopt` library[3], a Jax library for differentiable optimization. It relies on implicit differentiation to compute the derivative of the *solution of an optimization problem* w.r.t parameters of the problem. In particular, I implemented the following tools:

- Anderson acceleration for fixed point solving (Walker and Ni, 2011), with applications to Deep Equilibrium models (Bai et al., 2019), or to speed up solvers, since the optimum of an optimization problem is often the fixed point of the optimization algorithm itself.
- SGD with Armijo line search as described in Vaswani et al. (2019).
- OSQP solver: a general framework to solve convex quadratic programs with linear inequality constraints, where every linear operator can be formulated implicitly. It includes most of the features of the original's OSQP solver in GPU version (Schubiger et al., 2020).

More details on this framework are given in section 3.1.

1.4.3 Statement of authorship

The unpublished contributions to the thesis are exclusively my work. Some chapters are largely taken from published articles, for which I detail below the contributions.

Chapter 3. My contribution was the design and the implementation of a differentiable NMF in the following paper:

Thomas Fel, Agustin Picard, Louis Béthune, Thibaut Boissin, David Vigouroux, Julien Colin, Rémi Cadène, Thomas Serre, **CRAFT: Concept Recursive Activation FacTorization for Explainability**, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), *2023*

The other contributions are unpublished work.

Chapter 4. Most parts of this chapter have been published in:

L. Béthune, T. Boissin, M. Serrurier, F. Mamalet, C. Friedrich, and A. G. Sanz. **Pay attention to your loss : understanding misconceptions about Lipschitz neural networks.**, Advances in Neural Information Processing Systems (NeurIPS), *2022*.

The theoretical results are my own work, while the experimental results are joint work with Thibaut Boissin. Mathieu, Franck, Corentin, and Alberto played an important role in the discussion, writing, and proofreading.

Chapter 5. Most parts of this chapter have been published in:

Louis Béthune, Paul Novello, Thibaut Boissin, Guillaume Coiffier, Mathieu Serrurier, Quentin Vincenot, Andres Troya Galvis, **Robust One-Class Classification with signed distance function using 1-Lipschitz neural networks**, International Conference on Machine Learning (ICML), *2023*.

The theoretical results and the code are my work. The experimental results on concurrent baselines can be attributed to Paul and Guillaume. The original algorithm stems from the ideas of Mathieu. Quentin and Andres also took part in discussions of the early work.

[3] Distributed under Apache V2 license.

Chapter 6. The whole chapter have been published at:

Louis Béthune, Thomas Masséna, Thibaut Boissin, Yannick Prudent, Corentin Friedrich, Franck Mamalet, Aurelien Bellet, Mathieu Serrurier, David Vigouroux, **DP-SGD Without Clipping: the Lipschitz neural network way**, International Conference on Learning Representations (ICLR), *2024*.

The original ideas stem from a discussion between Aurelien and me. I supervised Thomas Massena as an intern, who played an important role in both theoretical and experimental results. Thibaut contributed to the code of the library, while Corentin and Yannick contributed to run some experiments. All authors were deeply involved in discussions, proofreading, or figure design.

Chapter 7. Parts of the work of this chapter come from the preprint:

González-Sanz, A., De Lara, L., Béthune, L. and Loubes, J.M.. **GAN estimation of Lipschitz optimal transport maps**, *2022*.

This chapter also presents some unpublished results related to Wasserstein-2 $\mathcal{W}_2$ distance and on the parametrization of convex functions and their gradient.

Appendix C. Some parts of this chapter have been published in the following papers, on which I coded and designed all the experiments. The theoretical work must be attributed to my co-authors.

François Bachoc, Louis Béthune, Alberto Gonzalez-Sanz, Jean-Michel Loubes, **Gaussian Processes on Distributions based on regularized optimal transport**, Artificial Intelligence and Statistics Conference (AISTATS), *2023*

François Bachoc, Louis Béthune, Alberto Gonzalez-Sanz, Jean-Michel Loubes, **Improved learning theory for kernel distribution regression with two-stage sampling**, arxiv preprint, *2023*

Finally, other contributions to the field are part of collaborations that are beyond the scope of this manuscript, and will not be presented here. Here is the list:

Thomas, FEL, Boutin, V., Moayeri, M., Cadene, R., Béthune, L., Andéol, L., Chalvidal, M. and Serre, T., 2023, November. **A Holistic Approach to Unifying Automatic Concept Extraction and Concept Importance Estimation.** In Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS), *2023*.

Serrurier, M., Mamalet, F., Fel, T., Béthune, L. and Boissin, T.. **On the explainable properties of 1-Lipschitz Neural Networks: An Optimal Transport Perspective**. In Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS), *2023*.

Mullor, T., Vigouroux, D. and Bethune, L.. **Efficient circuit implementation for coined quantum walks on binary trees and application to reinforcement learning.** In 2022 IEEE/ACM 7th Symposium on Edge Computing (SEC) (pp. 436-443). IEEE. *2022*.

Chapter 2

Lipschitz and orthogonal constraints in deep learning

In this chapter, we examine the existing literature regarding the parametrization of Lipschitz neural networks, and in particular *orthogonal* neural networks.

Contents

First, we define Lipschitz constrained neural networks.

Definition 6 (Lipschitz constrained feed-forward neural network)**.** *The space of feedforward neural network of depth D, with input space $\mathcal{X} \subset \mathbb{R}^n$, output space $\mathcal{Y} \subset \mathbb{R}^K$ (e.g logits), and parameter space $\Theta \subset \mathbb{R}^p$, is a parameterized function $f : \Theta \times \mathcal{X} \to \mathcal{Y}$ defined by the sequential composition of layers $f_1 \dots f_D$:*

$$f(\theta, x) := \left(f_D(\theta_D) \circ \dots \circ f_2(\theta_2) \circ f_1(\theta_1)\right)(x). \tag{2.1}$$

The parameters of the layers are denoted by $\theta = (\theta_d)_{1 \leq d \leq D} \in \Theta$. For affine layers, it corresponds to bias and weight matrix $\theta_d = (W_d, b_d)$. For activation functions, there are no parameters: $\theta_d = \varnothing$.

The space of Lipschitz-constrained neural networks is a subset of the space of feedforward networks, with the additional constraint that each layer

$$x_d \mapsto f_d(\theta_d, x_d) := y_d$$

is ℓ_d-Lipschitz for all θ_d. Consequently, the function $x \mapsto f(\theta, x)$ is ℓ-Lipschitz with

$$\ell \leq \ell_1 \times \ldots \times \ell_D$$

for all $\theta \in \Theta$.

In the following, LipNet1 will denote the class of Lipschitz-constrained networks with constant $\ell = 1$. For simplicity, all the networks in this manuscript are considered ℓ-Lipschitz with respect to $\|\cdot\|_2$ norm.

Remark 2.1. Lipschitzness w.r.t the inputs

Note that in this context the Lipschitz constraint is understood w.r.t to the *input* of the neural network, not w.r.t the *parameters*. Typically, Lipschitzness w.r.t the input is insufficient to ensure Lipschitzness w.r.t the parameters, even for linear models. However, under suitable assumptions on the input and the activations, the Lipschitz constant w.r.t parameters can also be bounded. This is explored in chapter 6.

2.1 Lipschitz constant estimation

The estimation of the Lipschitz constant is a hard task, known to be NP-hard since the seminal work of Scaman and Virmaux (2018). Therefore, on any neural network of meaningful size, one must resort to algorithms with imperfect estimations, or prohibitively high computation times. In the context of certification against adversarial attacks, the imperfect estimation must be an upper bound and not a lower bound.

2.1.1 Empirical methods for lower bounds

The most straightforward method is to evaluate the Lipschitz constant *empirically*. Those methods are *empirical* because they are obtained from a measure on some finite set $\mathcal{S} \subset \mathbb{R}^m$. There are several proxies of interest such as:

- the maximum of $\|\nabla_x f(x)\|_2$ over $\mathcal{S}$.
- the maximum of $\frac{\|f(x)-f(y)\|_2}{\|x-y\|_2}$ for all pairs $x, y \in \mathcal{S}$ such that $x \neq y$.
- the maximum of $\frac{\|f(x)-f(x+\delta)\|_2}{\|\delta\|_2}$ for $x \in \mathcal{S}$, with δ an adversarial attack returned by any attack method (preferably w.r.t l_2 norm).

Since the true Lipschitz constant ℓ is defined as a supremum over the whole domain $\mathbb{R}^d$, any method that uses a proxy over a subset $\mathcal{S} \subset \mathbb{R}^m$ yields a lower bound. This is not suitable for certification purposes but might be enough to give insights on training or generalization. We refer to the recent work of Khromov and Singh (2023).

Remark 2.2. Dirichlet energy

Some of these quantities are known under other names in literature, like "Dirichlet energy" (Dherin et al., 2022) for the average of $\|\nabla_x f(x)\|_2$. Therefore, the Lipschitz constant

can be seen as (yet!) another proxy to evaluate quantities of interest like generalization.

2.1.2 Formal methods for upper bounds

Certifications methods against adversarial attacks can be used to estimate Lipschitz bounds which motivates the (brief) survey of the section.

Virmaux and Scaman (2018) introduce the *AutoLip* algorithm, which compute Lipschitz bounds of a whole network by backpropagating bounds of each layer, based on the observations that if each f_d is ℓ_d-Lipschitz then their composition $f_d \circ \dots f_1$ is $\ell_d \times \dots \times \ell_1$-Lipschitz. We study a natural extension of their framework in chapter 6.

Libraries like Decomon (Airbus, 2023) or auto-LiRPA (Xu et al., 2020) provide tighter bounds for X_d via linear relaxations (Singh et al., 2019; Zhang et al., 2018a). Note that some of these methods are restricted to the computation of a *local* Lipschitz constant.

In Weng et al. (2018a) the authors propose an efficient method for ReLU networks by forward propagation of polyhedrons. There exist methods based on exactness verification (Ebihara et al., 2023). We can also mention the tools from extreme value theory (Weng et al., 2018b).

In convex relaxations, the initial problem is *relaxed* into a form amenable to convex optimization, which guarantees that the optimum can be found efficiently. Depending on the quality of the relaxation, the bounds may be more or less tight, and the solution will be found more or less quickly. A striking example of this approach is the **LipSDP** framework introduced in Fazlyab et al. (2019), later improved by Wang et al. (2022b). Note that this framework is very expressive and can handle a lot of activations functions, like GroupSort, as detailed in Pauli et al. (2023). We can also mention **LiPopt** (Latorre et al., 2019) among other candidates for certification methods.

2.2 Parametrizations of Lipschitz networks

This thesis does not address the parametrization of Lipschitz networks: instead, the study focuses on the class of Lipschitz functions as a whole. However, for practical implementation, we detail in this section the major contributions of the field related to the parametrization of Lipschitz functions. The reader in a hurry can just read section 2.2.1 and skip the rest.

2.2.1 Universal approximation

In practice, this is enforced by using activations with Lipschitz constant l_d, and by applying a constraint $\Pi : \mathbb{R}^p \to \Theta$ on the weights of affine layers. This corresponds to spectrally normalized matrices (Yoshida and Miyato, 2017; Bartlett et al., 2017), since for affine layers we have $l_d = \|W_d\|_2 := \max_{\|x\|_2 \leq 1} \|W_d x\|_2$ hence $\Theta = \{W_d, \|W_d\| \leq l_q\}$. Note that $\|W_d\|_2$ denotes the spectral norm, while $\|x\|$ and $\|W_d x\|_2$ are (vector) euclidean norm. The parametrization of Lipschitz networks is detailed in the next section.

The seminal work of Anil et al. (2019) proved that universal approximation in the set of l-Lipschitz functions was achievable with certain constraints on the weights, and with the GroupSort activation function. GroupSort activation has the specificity to operate on pairs of consecutive neurons - it is **not** an *elementwise* activation, which contrasts with other common activation functions. It is defined as follow for all coordinates i:

$$\text{GroupSort2}(x)_{2i,2i+1} = [\min(x_{2i}, x_{2i+1}), \max(x_{2i}, x_{2i+1})]. \tag{2.2}$$

The theorem of Anil et al. (2019) bounds the "two-to-infinity" norm (Cape et al., 2019) of the first layer, noted $\|\cdot\|_{2\to\infty}$, and the $\|\cdot\|_\infty$ norms of further layers to obtain universal approximation in $\mathrm{Lip}_1(\mathcal{X},\mathbb{R})$. These networks are 1-Lipschitz by construction.

Remark 2.3. Universal Approximation in the space of L-Lipschitz functions

LipNet1 networks also benefit from an universal approximation theorem in $\mathrm{Lip}_1(\mathcal{X},\mathbb{R})$ with respect to uniform convergence Anil et al. (2019). Note that

$$\mathrm{Lip}_L(\mathcal{X},\mathbb{R}^K) = \{Lf \mid f \in \mathrm{Lip}_1(\mathcal{X},\mathbb{R}^K)\} \tag{2.3}$$

so LipNet1 can be used to approximate functions in $\mathrm{Lip}_L(\mathcal{X},\mathbb{R}^K)$.

In practice, authors of Anil et al. (2019) reported that bounding spectral norm $\|\cdot\|_2$ and enforcing orthogonality of rows/columns of weight matrices (i.e $W_i^T W = I$) yielded the best empirical results because it turned the network into a *Gradient Norm Preserving* network. Orthogonal networks benefit from their own field of research, as detailed in Section 2.3.

Most activation functions are Lipschitz, the popular including ReLU, sigmoid, tanh, softplus, etc. Self-Attention is not Lipschitz (Kim et al., 2021), but solutions have been proposed in Xu et al. (2022a) and Qi et al. (2022). In Zhai et al. (2023) the update of the spectral norm and of the weight itself are decoupled. Lipschitz recurrent units have been proposed in Helfrich et al. (2018) and Erichson et al. (2021).

While there exist numerous approaches for the parametrization of Lipschitz networks, they mainly rely on optimization over matrix manifolds (Absil et al., 2009). For example, we can mention differentiable re-parametrization (aka "trivialization") like Miyato et al. (2018); Anil et al. (2019), direct parametrization (Meunier et al., 2022b; Wang and Manchester, 2023), or projections (Arjovsky et al., 2017). We highlight below two notable strategies:

1. In **differentiable reparametrization** $\Pi : \mathbb{R}^p \to \Theta$ where $\tilde{\theta} = \Pi(\theta)$: the weights $\tilde{\theta}$ are used during the forward pass, but the gradients are back-propagated to θ through Π. This turns the training into an unconstrained optimization problem on the landscape of the loss $\mathcal{L} \circ f \circ \Pi$.
2. With a suitable **projection operator** $\Pi : \mathbb{R}^p \to \Theta$: this is the celebrated Projected Gradient Descent (PGD) algorithm (Bubeck et al., 2015) applied on the landscape of the loss $\mathcal{L} \circ f$.

2.2.2 Regularization and penalties

Regularization approaches in general do not give formal guarantees, only a very crude upper bound. Indeed, the optimizer strives for a balance with the objective and the regularization, and in some circumstances the local Lipschitz constant may blow up.

The Lipschitz constant of affine layers can be constrained with a Gradient penalty (Gulrajani et al., 2017), a regularization term taking the form $(1 - \|\nabla_x f\|_2 - 2)^2$. It has the disadvantage of requiring nested differentiation of f: once with w.r.t x, and another one with w.r.t θ. This can be costly. This technique is also used in Zhou et al. (2019) for Lipschitz GAN.

Spectral regularization (Yoshida and Miyato, 2017) operates on the weights themselves, unlike gradient penalty which operates on the whole network. Among concurrent approaches based on regularization we can also mention Cisse et al. (2017) or Gouk et al. (2021).

2.2.3 Constraints and clipping

In the seminal Wasserstein-GAN article (Arjovsky et al., 2017) the authors propose the "weight clipping" operation, which consists in projecting each coefficient W_{ij} onto the $[-c, +c]$ interval. Another related strategy is the "Frobenius normalization" (Salimans and Kingma, 2016), i.e. the projection $W \mapsto \frac{W}{\|W\|_F}$, or the spectral normalization (Miyato et al., 2018) with $W \mapsto \frac{W}{\|W\|_2}$. These approaches lead to a tighter upper bound than regularization. As all norms are equivalent in finite dimension, all those methods are related to each other, with multiplicative constants that depend on the size of the matrix. Naively stacking such layers may lead to vanishing gradients. Indeed, projections and clipping ensure $\sigma_{\max} \leq K$ for some K, but the other singular values of W may collapse to zero.

Spectral normalization requires the leading singular value, which can be computed with Power Iteration[1] (Mises and Pollaczek-Geiringer, 1929) or SVD decomposition (Trefethen and Bau, 2022). Note that typical implementations of SVD decomposition are not well-behaved during back-propagation and may suffer from numerical issues, which have been partially addressed in Wang et al. (2019a).

In some applications (like robustness certification) the estimate of the Lipschitz constant must be an *upper bound*, and not a *lower bound*. Unfortunately, Power Iteration solves a maximization problem: the leading singular value is estimated *from below*. Therefore, incomplete convergence may yield poor estimates of the true Lipschitz constant. Moreover, on big matrices or ill-conditioned matrices, waiting for convergence might be impracticable. This led Delattre et al. (2023b) to propose a new method estimating the leading singular value *from above*. Moreover their method, like Power Iteration, only relies on matrix-vector products so they can be applied to any (sparse) linear operator like convolutions. This is also the direction chosen by the recent work of Ebrahimpour-Boroojeny et al. (2023), which furthermore relies on AutoDiff to extract the matrix-vector product without relying on the matrix.

2.2.4 An emerging approach: direct parametrizations

Another line of work was opened by the seminal works Meunier et al. (2022b) and Araujo et al. (2022). This approach contrasts with other methods: instead of constraining each layer to be 1-Lipschitz, a whole *block* of layers is made 1-Lipschitz by reasoning at the scale of a whole block. However, universal approximation results are still lacking with this family of architectures.

Discretization of dynamical systems

Following the line of work initiated by Haber and Ruthotto (2017); Lu et al. (2018); Chen et al. (2018), Meunier et al. (2022b) and Araujo et al. (2022) re-interpret the forward pass in a residual network as the discretization of a continuous dynamical system. This approach allows reasoning on the properties of the dynamical system, for which enforcing the constraints is easier. More precisely, they show that for the dynamical system

$$\begin{cases} x_0 & \in \mathcal{X}, \\ \frac{\partial x_t}{\partial t} & := F_t(x_t) := -\nabla_x f_t(x_t) + A_t x_t \end{cases} \tag{2.4}$$

with f_t convex, and A skew-symmetric, satisfies for every x_0, z_0 the property $\|x_0 - z_0\|_2 \leq \|x_t - z_t\|$ holds for all $t > 0$. The remaining difficulty is to ensure that the properties are preserved under a

[1] Power Iteration will celebrate its centenary in 2029, don't miss it.

suitable discretization scheme. By choosing the mid-point Euler method and a clever parametrization of the convex function f_t, they show that their method ends up as:

$$z = x - \frac{2}{\|W\|_2^2} W^T \sigma(Wx + b), \tag{2.5}$$

with σ a 1-Lipschitz activation function. Interestingly, other knowns Lipschitz layers such as Cayley convolutions or Householder activation can be retrieved as a special case of the above framework (see section 2.3 for more details).

Formulations based on matrix optimization problems

In Wang and Manchester (2023) the bounds on the Lipschitz constant are computed using a quadratic program. Interestingly, this formulation allows to bound the Lipschitz constant, if the coefficients (and structure) appearing in the quadratic program are controlled. For a clever choice of structure, not only the Lipschitz constant of the operator is bounded, but its implementation takes the form of a "sandwich" layer. This sandwich layer is a (highly structured) composition of other layers with some constraints on weight matrices and activation functions.

2.3 Gradient Norm Preserving ("Eikonal") networks

LipNet1 networks fulfilling $\|\nabla_x f(x)\| = 1$ almost everywhere wrt any intermediate activation x are said to be Gradient Norm Preserving (GNP), and elegantly avoids the vanishing gradients phenomenon (Li et al., 2019a; Bansal et al., 2018). They are also present in WGAN training (Gulrajani et al., 2017). This property is typically achieved in affine layers with orthogonal matrices, which justify the *"orthogonal neural network"* terminology (Stasiak and Yatsymirskyy, 2006; Li et al., 2019b).

Definition 7 (Gradient Norm Preserving Networks)**.** *GNP networks are 1-Lipschitz neural networks with the additional constraint that the Jacobian of layers consists of orthogonal matrices:*

$$\left(\frac{\partial f_d}{\partial x_d}\right)^T \left(\frac{\partial f_d}{\partial x_d}\right) = I. \tag{2.6}$$

For example, this is achieved with GroupSort activation (Anil et al., 2019; Tanielian and Biau, 2021), Householder activation (Mhammedi et al., 2017), and orthogonal weight matrices (Li et al., 2019a,b) or orthogonal convolutions (see Achour et al. (2022); Singla and Feizi (2022); Xu et al. (2022b) and references therein). Without biases these networks are also norm preserving: $\|f(\theta, x)\| = \|x\|$.

However, to this day, a universal approximation of Lipschitz function by the mean of orthogonal networks is still lacking.

Warning 2.1. Expressiveness of orthogonal networks.

It is not clear why orthogonal matrices help in reaching maximum accuracy. Solving the vanishing gradient phenomenon that plagues spectrally normalized matrices is an element of response, but it does not tell the full story. The Eikonal property $\|\nabla_x f\|_2 = 1$ might be extremely severe for some applications and induces a bias. This bias is either explicit (i.e. intrinsic limitations of this family of architectures) or implicit (i.e. bad interactions with the stochastic optimizer). It may generalize in unexpected ways that may be, or may not be,

aligned with "real-world data". One heuristic argument in this direction is the following: if one wants to parametrize a function with slope $s < 1$, the Eikonal network can do so by using "triangular stairs", defined by a "step function" $S(x) = x\mathbb{1}_{x\leq a} + (a - x)\mathbb{1}_{x>a}$ with $x \in [0, 1]$ and $a = (1/2)(1 + s)$. This base step function S can be re-scaled as $\hat{S}_w(x) = wS(x/w)$ to generate steps of width w with $x \in [0, w]$. The steps are glued together (to ensure continuity), and by taking the limit $w \to w$ we obtain an approximation of a slope s with stairs of slope 1. We see that the number of steps grows polynomially with $\frac{1}{w}$. Eikonal networks are piecewise affine, and the number of pieces typically depends on the number of neurons (and depth!). Therefore, approximating a slope s with Eikonal network requires a lot of neurons, whereas a single one is necessary with conventional networks.

The following section is devoted to a short literature review on orthogonal layers, that the reader in a hurry may skip.

2.3.1 Orthogonal layers

Anil et al. (2019) establish that GNP networks with ReLU are exactly affine functions. They proposed Sorting activation functions to circumvent the expressiveness issue. In particular, GroupSort2 revealed to be an efficient alternative (Tanielian and Biau, 2021) to ReLU, and can be seen as a particular case of Householder reflections (Mhammedi et al., 2017; Singla et al., 2021). Other authors tried to fix ReLU itself (Huang et al., 2021). Note that layers like "group normalization" or "layer normalization" are not Lipschitz (because of the division occurring). However, layer centering is 1-Lipschitz and almost orthogonal.

Property 3. Bounded loss gradient for layer centering. *Layer centering is defined as* $f(x) = x - (\frac{1}{n}\sum_{i=1}^{n} x_i)\mathbf{1}$ *where* $\mathbf{1}$ *is a vector full of ones and acts as a "centering" operation along some channels (or all channels). Then the singular values of this linear operation are:*

$$\sigma_1 = 0, \quad \textit{and} \quad \sigma_2 = \sigma_3 = \ldots = \sigma_n = 1. \tag{2.7}$$

In particular $\|\frac{\partial f}{\partial x}\|_2 \leq 1$.

The proof is given in appendix B.

In Prach and Lampert (2022) the authors leverage certain structure on the matrix to obtain an *almost orthogonal* layer, that can also be extended to convolutions.

2.3.2 Optimization over the Stiefel manifold

The literature on the topic is plethoric. It has been extensively studied in Absil et al. (2009), while Arjovsky et al. (2016); Hyland and Rätsch (2017); Lezcano-Casado and Martınez-Rubio (2019); Huang et al. (2018) focus on neural networks retractions like Cayley transform or the exponential map; more recently Ablin and Peyré (2022) proposed a landing algorithm, and Kerenidis et al. (2021) proposed an algorithm inspired by quantum computing, while Choromanski et al. (2020) proposed an approach based on graph matching.

The set of orthogonal matrices, and its generalization the Stiefel manifold (Absil et al., 2009), are not convex, and not even connected. This makes the optimization over these sets challenging.

In this section, W denotes the matrix of the linear operator, $C \in \mathbb{N}$ the number of input channels, and $M \in \mathbb{N}$ the number of output channels. $SO(C)$ denotes the special orthogonal group of matrices of size $C \times C$. This is a connected Lie group that consists of all the orthogonal matrices whose

determinant is 1. $O(C)$ denotes the orthogonal group. This is a Lie group with two connected components, having $\{-1, +1\}$ determinants. $St(M, C)$ denotes the Stiefel manifold, defined as the set of orthonormal M-frames in $\mathbb{R}^C$, i.e. the set of ordered orthonormal M-tuples of vectors in $\mathbb{R}^C$.

Remark 2.4. Three flavors of orthogonal transformations.

The following inclusions hold

$$\begin{array}{c}\text{Special}\\ \text{Orthogonal Group}\\ \text{M=C}\\ \det=+1\end{array} \subsetneq \begin{array}{c}\text{Orthogonal Group}\\ \text{M=C}\\ \det \in \{-1/+1\}\end{array} \subsetneq \begin{array}{c}\text{Stiefel Manifold}\\ M \neq C\end{array} \tag{2.8}$$

Because of width constraints in networks, most works attempt to optimize over the Stiefel manifold, as the orthogonal group requires exclusively square matrices. Some methods only work on the special orthogonal group.

One typically look for methods allowing optimization in the outermost set, the Stiefel manifold. The dimension of Stiefel manifold is $MC - \frac{1}{2}C(C+1)$, where we assumed $M > C$ (see Chap 3. p26 in Absil et al. (2009)). When the matrix is square it simplifies into $\frac{C(C-1)}{2}$, since the tangent space of the Stiefel manifold consists of skew-symmetric matrices, whose dimension is indeed $\frac{C(C-1)}{2}$.

Differentiable re-parametrizations ("trivializations")

In this setting the constrained optimization problem is re-casted as an unconstrained optimization problem with a different optimization landscape:

$$\arg\min_{x \in \mathbb{R}^p} F(\Pi(x)) \tag{2.9}$$

where $\Pi : x \mapsto \mathcal{K}$ maps an arbitrary parameter vector $x \in \mathbb{R}^p$ of dimension p onto the Stiefel manifold (or Special Orthogonal, or Orthogonal groups). Such mapping can be surjective (all orthogonal matrices can be represented), bijective (the mapping is one-to-one), or with looser properties (i.e. a parameterization can fail to represent all matrices, some re-parametrizations are redundant). As noticed in Warning 1, it is not clear which property would yield *a priori* the best results in the context of deep learning. See Lezcano Casado (2019) and references therein for this family of methods. Different candidates for Π are detailed below.

Gram-Schmidt (GS) and Modified Gram-Schmidt (MGS)

These algorithms compute the QR factorization of the matrix. We are left with the orthogonal matrix Q. MGS is notoriously more stable than GS (Greenbaum et al., 1997).

The Cayley transform and the exponential map are two related methods that relies on a bijection between the special orthogonal group and skew-symmetric matrices, i.e. matrices A that fulfill $A + A^T = 0$. The projection of any matrix W onto the set of skew-symmetric matrices is given by $\frac{1}{2}(W - W^T)$: this transformation comes in handy when a skew-symmetric matrix is required.

Cayley Transform

Cayley transform is based on the mapping

$$Q = (I - A)(I + A)^{-1}. \tag{2.10}$$

Algorithm 1 QR factorization by Modified Gram-Schmidt

Require: $W = [w_i]_{i\in[0,C-1]} \in \mathbb{R}^{C\times C}$,
Ensure: $W = QR$ with $Q \in O(C)$ and $R \in \mathbb{R}^{C\times C}$ an upper-triangular matrix
  $w_i^c = w_i,\ \forall i \in [0, C-1]$ ▷ Copy of input matrix
  for $j = 1$ to C **do**
    $r_{jj} = ||w_j^c||_2$
    $q_j = w_j^c / r_{jj}$
    for $k = j+1$ to C **do**
      $r_{jk} = q_j^T w_k^c$
      $w_k^c = w_k^c - r_{jk} q_j$
    end for
  end for
  return QR

The extension to rectangular matrices is described in Macías-Virgós et al. (2018), and corresponds to Algorithm 2. Cayley transformation is also the Padé approximant of degree $(1, 1)$ of the exponential map (Lezcano-Casado and Martınez-Rubio, 2019), see below.

Algorithm 2 Computing Cayley transform

Require: $W \in \mathbb{R}^{M\times C}$, case $M > C$
Ensure: $\hat{W} \in \mathbb{R}^{M\times C}$ is an orthogonal matrix
  $U, V = W[:C, :], W[C:, :]$ ▷ $U \in \mathbb{R}^{C\times C}$, $V \in \mathbb{R}^{M-C\times C}$
  $A = U - U^T + V^T.V$ ▷ Remark: A is not skew-symmetric
  $Z = (I + A)^{-1}$
  $\hat{W}_1 = Z(I - A)$ ▷ $\hat{W}_1 \in \mathbb{R}^{C\times C}$
  $\hat{W}_2 = -2VZ$ ▷ $\hat{W}_2 \in \mathbb{R}^{M-C\times C}$
  $\hat{W} = [\hat{W}_1, \hat{W}_2]$ ▷ $\hat{W} \in \mathbb{R}^{M\times C}$
  return $\hat{W}$

Exponential map

If A is skew-symmetric, then

$$\Theta = \exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!} \text{ is orthogonal.} \tag{2.11}$$

This transformation only represents the special orthogonal group $SO(n)$. Moreover, since the exponential map of a connected compact Lie group is always surjective, it turns out that every orthogonal matrix with unit determinant can be written as the exponential of some skew-symmetric matrix. Approximating the matrix exponential can be performed with various methods. Taylor expansion and Padé approximation (Baker Jr and Gammel, 1961) are often used. Note that Taylor expansion truncated to degree k is the Padé approximant of order $(p, q) = (k, 0)$.

Product of Householder matrices

According to a result of Uhlig (2001), every $C \times C$ orthogonal matrix can be decomposed into a product of Householder matrices:

$$U = \prod_{i=1}^{C} H_i, \qquad \text{with } H_i = I - 2\frac{v_i v_i^T}{\|v_i\|_2^2}, v_i \in \mathbb{R}^C \tag{2.12}$$

where $v_i v_i^T$ is an *outer product* returning a $C \times C$ matrix of rank 1, and where H_i is a *Householder matrix*, which has the property of being orthogonal. This decomposition is stable under gradient descent updates Mhammedi et al. (2017) which makes it a valid re-parametrization. This can be seen as a special case of the QR factorization when $R = I$. Different strategies are available to efficiently compute the product:

1. **Naive product.** Each product of H_i with H_{i+1} costs $\mathcal{O}(C^3)$. Chaining those C products induces a $\mathcal{O}(C^4)$ cost to form U.
2. **Matrix-vector products (see Mhammedi et al. (2017)).** We are usually interested in computing the matrix-vector product Ux for some $x \in \mathbb{R}^C$. In this case the product is rewritten $Ux = H_1 \dots (H_{C-1}(H_C x))$. Each $H_i x = x - v_i \frac{v_i^T x}{\|v_i\|_2^2}$ product only cost $\mathcal{O}(C)$ because $v_i^T x$ is an inner product between two vectors of dimension C. The total cost falls to $\mathcal{O}(C^2)$. Hence the parametrization is "free" in the sense that it is no more expensive than computing Ux directly from U. Observe that $U = J_X(X \mapsto UX)$ so Autodiff can be used to retrieve U from VJP (vector-jacobian products) automatically.
3. **Partial parallelization.** Due to its sequential nature of C products, the algorithm cannot be efficiently parallelized on GPU. In Mathiasen et al. (2020) the authors propose a novel algorithm, in which the overall cost remain $\mathcal{O}(C^2)$ but with only $\frac{C}{b} + b$ sequential products, for some $1 < b < C$ that divides C. Theory suggests to use $b = \sqrt{C}$, while in practice the optimal value is selected with a clever exhaustive search. They report a $\times 29$ speed-up increase. However this implementation increases the complexity of the code by a non negligible factor.

See Zhang et al. (2018b) for additionnal insights on Householder parametrization.

Differentiable Projections

This instance is a special case of differentiable re-parametrization (see section 2.3.2). In this case, input and output spaces share the same dimension. We look for a projector $\Pi : \mathbb{R}^{M \times C} \to St(M, C)$ onto the Stiefel manifold:

$$\begin{aligned} \mathcal{K} := \Pi(W) = \arg \min_{\Theta \in \mathbb{R}^{M \times C}} & \frac{1}{2}\|W - \Theta\|^2 \\ \text{such that } & \Theta^T\Theta - I = \mathbf{0}. \end{aligned} \tag{2.13}$$

Since there is multiple connected component, the projection operator cannot be continuous everywhere. The feasible set $\Theta^T\Theta - I = \mathbf{0}$ is not convex either. The norm $\|\cdot\|$ can denote either the Frobenius norm

$$\|M\|_F^2 := \sum_{ij} M_{ij}^2 = \sum_{i=1}^{\min(m,n)} \sigma_i^2(M) = \text{Tr}(M^T M) \tag{2.14}$$

where σ_i is the i-th singular value of M, either the spectral norm $\|M\|_2^2 = \sigma_{\max}(M)$. The two formulations happen to be equivalent. Projections are also useful for Projected Gradient Descent (PGD) algorithms. Some examples of projection algorithms are detailed below.

Singular Value Decomposition

Let $W = U\Sigma V^T$ be the Singular Value Decomposition (SVD) of W, with $U^TU = I$, $V^TV = I$ and Σ the diagonal matrix containing the singular values. Then it is well known that $\Theta = \Pi(W) = \Pi(U\Sigma V^T) := UV^T$ is the orthogonal projection of W onto the Stiefel manifold. The overall cost of the SVD decomposition is typically $\mathcal{O}(MC\min(M, C))$. See Trefethen and Bau (2022) for more details.

Björck Projection

This method introduced in Björck and Bowie (1971) solves the optimization problem of equation (2.13) with an iterative algorithm that only involves matrix products. Here we detail the first order form of Björck algorithm. The sequence starts from $\Theta_0 \leftarrow \frac{W}{\sigma_{\max}(W)}$ and is expanded as follows:

$$\Theta_{t+1} \leftarrow \frac{3}{2}\Theta_t - \frac{1}{2}\Theta_t\Theta_t^T\Theta_t. \tag{2.15}$$

Note that this iteration corresponds to a step of the gradient descent for minimizing the loss $||\Theta^T\Theta - I||$, as detailed in section 2.3.2.

The estimation of $\sigma_{\max}(M)$ can be done with the methods discussed in section 2.2.3. As explained previously, Power Iteration tends to underestimate the true spectral norm when run for insufficient iterations. Fortunately, according to Björck and Bowie (1971) the convergence of this first order scheme is guaranteed as long as $\|W\|_2 < \sqrt{3}$, which leaves room for errors, while the convergence of the ∞-order scheme (not implemented in practice) is guaranteed as long as $\|W\|_2 \leq 1$. In practice, we can run the algorithm until the residuals $\|I - \Theta^T\Theta\|$ fall below some threshold ϵ or for a fixed number of iterations $t \in \mathbb{N}$ (hoping for the best). Note that the $\Theta_t\Theta_t^T\Theta_t$ product can be re-ordered as $(\Theta_t\Theta_t^T)\Theta_t$ or $\Theta_t(\Theta_t^T\Theta_t)$ whether $M < C$ or $M > C$. Hence each iteration induces a $\mathcal{O}(MC\min(M, C))$ cost. Such a method is also reported in Kovarik (1970).

Björck algorithm is the default method advocated in Anil et al. (2019) for the parametrization of Lipschitz networks, and this is also the default method implemented in **deel-lip** library. Some tests suggested than 15 iterations was more than enough for most use-case, including matrices of size 256×256 or bigger.

Orthogonalization by Newton's Iteration (ONI)

The following method is described for $M \leq C$ (row orthogonality), but it can be easily extended to the case $M > C$. The problem of eq. (2.13) has a closed-form solution, obtained from adapting the SVD decomposition:

$$\Theta = \left(WW^T\right)^{-\frac{1}{2}} W. \tag{2.16}$$

Solving (2.16) requires the computation of $\left(WW^T\right)^{-\frac{1}{2}}$ which can be performed using Newton's iteration (Huang et al., 2020). The authors called their method ONI, standing for *Orthogonalization by Newton's Iteration*. The Newton's method can be used to compute the inverse p-th root $A^{-\frac{1}{p}} \in \mathbb{R}^{M\times M}$ According to Bini et al. (2005), when initializing $X_0 \leftarrow I_M$, the sequence X_k defined recursively as

$$X_{k+1} \leftarrow \frac{1}{2}\left(3X_k - X_k^3A\right) \tag{2.17}$$

converges to $A^{-\frac{1}{2}}$. Convergence is ensured if the spectral radius of A is not greater than 1. Authors reported that running run 2 to 5 iterations of ONI was sufficient for most applications.

Implicit differentiation of the projection.

The problem (2.13) is not convex everywhere because the feasible set $\Theta^T\Theta - I = \mathbf{0}$ is not convex. The Stiefel manifold has two connected components, even in the $M = C$ case, because it contains the matrices of determinants $+1$ and -1. Note that det is a continuous application, hence having disjoint images $\{-1, +1\}$ proves that there are at least two disjoint components in its pre-image.

Conjecture 1. *The problem of Equation 2.13 is convex almost everywhere: the operator* Π *is continuous and differentiable almost everywhere. Therefore we can apply the implicit function theorem to compute derivatives for almost all values of* W*.*

The application of implicit function theorem yields a simple form for the derivative $\frac{\partial \Pi}{\partial W}$ (only requires solving a linear system with a symmetric matrix), making it amenable to fast computation. To the best of my knowledge, this method was not explored before my work. All the details are given in section 3.2.

Differentiation through unrolled iterations.

Björck algorithm, and power iteration, are two algorithms whose iterations are differentiable. Some *Tensorflow* or *Jax* implementations of SVD are also differentiable (often w.r.t Σ, less often w.r.t U, W^T factors). Finally, iterations of Newton method are also differentiable, see Annex A in Huang et al. (2020). Hence it is possible to obtain the derivative $\frac{\partial \Pi}{\partial W}$ from backpropagation to perform gradient steps on the unconstrained matrix W directly, circumventing the problem of computing Riemannian gradients over the Stiefel manifold. Thanks to Autodiff frameworks, the computation of the derivative is "free" and does not require additional coding effort. The backward pass typically costs about $\times 3$ times the runtime of the forward pass, because of the identities $(fg)' = fg' + f'g$ and $(f \circ g)' = g'(f' \circ g)$ that typically produces computation graphs three times bigger.

In Golinski et al. (2019), the authors report that the Björck algorithm may be unstable during the computation of gradients with unrolling. The tradeoff between implicit differentiation, or backpropagation through unrolled steps, is a hot topic that has gathered much interest in recent years, see for example Scieur et al. (2022) and references therein.

Projected Gradient Descent

In this scheme, the differentiability of Π is irrelevant. The gradient step is performed on the constrained matrix Θ directly and ends up *outside* the manifold. This algorithm is very simple and benefits from numerous studies. However, it does not benefit straightforwardly from momentum since the curvature of the manifold must be taken into account. Curvature-compatible algorithms are handled by Riemannian Gradient Descent (Bonnabel, 2013) and parallel transport Alimisis et al. (2021).

ProjUNN-D

The method introduced in Kiani et al. (2022) relies on the closed form of the projection onto the Stiefel manifold, based on the polar transformation given in Equation 2.16. To reduce the complexity of this projection, they propose to restrict the gradient update to a low-rank matrix G_k. The complexity is in $\mathcal{O}(k(C^2 + Ck + k^2))$, and the authors also discuss the numerical stability of this projection.

Riemannian Gradient Descent

In this setting we fall back to the original formulation:

$$\arg\min_{\mathcal{K}\in St} F(\mathcal{K}). \tag{2.18}$$

We are still relying on gradient-based optimization, but instead of using the Euclidean gradient $\nabla_x F$, the goal is to use the *Riemannian* gradient $\text{grad}_x f$, which is a vector living in the tangent space $T_{St}(x)$ at x in manifold St. Riemannian gradient coincides with Euclidean one when the manifold $\mathcal{M}$ is a vector space. We refer to Absil and Malick (2012) for more details on Riemannian optimization on matrix manifold.

Remark 2.5. Retractions.

Retraction is a central tool in Riemannian optimization. According to Absil and Malick (2012): "*Retractions generate approximations of geodesics that are first-order accurate. A retraction can also be viewed as providing "locally rigid" mappings from the tangent space into the manifold*". Most projections fulfill this definition. However, the opposite is false: some retractions are not projections (Lezcano-Casado and Martınez-Rubio, 2019). Recently, a "landing" algorithm Ablin and Peyré (2022) has been proposed, which get rid of the need for retractions.

ProjUNN-T

Like projUNN-D, the projUNN-T method (Kiani et al., 2022) computes the gradient G w.r.t. the loss. The gradient is then projected onto the tangent space of the Stiefeld Manifold with the operator $\Pi_{T_U}(X) = \frac{1}{2}(X - UX^TU)$. The initial matrix is transported or rotated in the direction of the projected gradient, using the exponential map.

$$U \mapsto U \exp\left(-\eta U^T \Pi_{T_U}(X)\right). \tag{2.19}$$

To reduce the computational complexity, authors propose to restrict the gradient update to a low-rank matrix. The complexity is also in $\mathcal{O}(k(C^2 + Ck + k^2))$.

Regularization for orthogonality constraint

In this approach, the constraint is loosely enforced by transformation into a regularization term $\phi : \mathbb{R}^{mn} \to \mathbb{R}$ that measures the "closeness" between the candidate W and the Stiefel manifold. We usually chose $\phi(\Theta) = 0$ when Θ is orthogonal.

$$\arg\min_{W\in\mathbb{R}^{mn}} F(W) + \lambda\phi(W). \tag{2.20}$$

See the work of Xiao and Liu (2021) for a list of penalty function useful to optimize over the Stiefel manifold.

Parseval's tightness

In Cisse et al. (2017) the authors propose to optimize the so-called *Parseval tightness* Kovačević et al. (2008) the weight matrices:

$$\Phi(W) = \frac{1}{2}\|I - W^T W\|_F^2. \tag{2.21}$$

The (matrix) derivative of Φ takes a simple form (see appendix B. from Anil et al. (2019)):

$$\nabla_W \Phi(W) = -W + WW^T W. \tag{2.22}$$

The gradient descent with stepsize λ on the regularized objective of equation 2.21 yields:

$$W_{t+1} \leftarrow W_t - \lambda(-W_t + W_t W_t^T W_t) = (1+\lambda)W_t - \lambda W_t W_t^T W_t.$$

Observe that when $\lambda = 0.5$ we recognize the first order update rule of Björck algorithm.

Parseval's tightness with Matrix-vector products.

The aforementioned methods can suffer from a high cost for a high dimensional W. This is typically the case for the Toeplitz matrix corresponding to a convolution (Araujo et al., 2021). Fortunately, it is possible to adapt it in a stochastic variant that only requires matrix-vector products:

$$\phi(W) = \mathbb{E}_{x\sim\mathbb{R}^n}[\frac{1}{2}\|x - W^T(Wx)\|_2^2]. \tag{2.23}$$

This makes the method amenable to convolutions for which the operations $x \mapsto Wx$ and $x \mapsto W^T x$ can be computed efficiently.

Mixing approaches

To add to the confusion, those approaches are not necessarily mutually exclusive and it is possible to combine regularization (see section 2.3.2 with projected gradient steps (see section 2.3.2), or Riemannian gradient steps 2.3.2. It is also possible to combine differentiable projections (see section 2.3.2) and apply a PGD step every T regular gradient steps (GD). To this day, a comprehensive benchmark is lacking to determine the best orthogonalization algorithm in deep learning.

Warning 2.2. Metrics that must be monitored for a comprehensive benchmark.

Optimization in deep learning is always a complicated question because there are typically two quantities in tension. On one hand, one desires an efficient optimizer that navigates the optimization landscape and finds the best (local) optimum. On the other hand, finding the optimum on the train set is by no means a guarantee that the test loss will be low. Some "inefficiency" of the optimizer might push it away from a local optimum that overfits toward a local optimum that generalizes, *even though its train error is higher*. This interlacing is typical of deep learning and makes every attempt to improve the optimizer tedious. The following metrics are of special interest in balancing everything:

- Time/Space Complexity.
- Wallclock Runtime on CPU *and* GPU/TPU.
- Memory consumption, maximum network size supported on typical hardware.
- Numerical accuracy in *float32* and *float64* environments.
- Compatibility with stochastic optimization or momentum.
- Metric on the final task, *including on the test set*, which can be accuracy or even optimal transport maps.

The last point is useful to monitor if the optimizer leverages *an implicit bias* beneficial to

the final task. This was initially intended as a future work but the very recent benchmark of Prach et al. (2023) answers (partially) the question.

2.3.3 Orthogonal convolutions

Strict orthogonality is challenging to enforce, especially for convolutions for which it is still an active research area: see Trockman and Kolter (2021); Singla and Feizi (2021b); Achour et al. (2022); Singla and Feizi (2022); Xu et al. (2022b) and references therein.

The "truly orthogonal" convolutions have been characterized by the work of Achour et al. (2022): this requires a circular padding, and the number of channels must be related to each other and depends on the image size. In this work, the authors build upon the regularization proposed in Wang et al. (2020) to achieve orthogonality.

Some concurrent work pretends to parametrize orthogonal convolutions. But most of the time the operator is only *almost orthogonal*, or it does not exhibit the structure of a true convolution anymore. Almost orthogonal convolutions typically have some singular values strictly smaller than 1, and less frequently bigger than 1. This may induce vanishing gradients during back-propagation, if the cotangent vector falls in the (almost) null space of the operator. When the operator loses its convolution structure, it may lose some of its sparseness properties in pixel space. It cannot be efficiently implemented with Deep learning frameworks, and it must be performed in Fourier space. This induces a higher cost than conventional (unconstrained) convolutions. Whether or not strict orthogonality is a desirable property for learning is a question that remains to be answered.

In this setting $\mathcal{K}$ denotes both the linear operator and the matrix that operates on vectorized (i.e. flattened) input tensors.

$H, W \in \mathbb{N}$	Height and width inputs.
$C \in \mathbb{N}$	Input channels.
$M \in \mathbb{N}$	Output channels.
$S \in \mathbb{N}$	Stride.
$k, l \in \mathbb{N}$	Kernel size. Often $k = l$.

For convolution $n = k^2CHW$, $m = M(H/S)(W/S)$, and $m \leq n$ is not equivalent to $n \leq m$. Only circular padding ensures the existence of truly orthogonal convolutions (Achour et al., 2022).

Structure of convolutions as linear operators

For a convolution operator $\mathcal{K} \star x$, the image x can seen as flattened vector $\text{flat}(x)$, and the corresponding matrix Θ can be studied. It appears that such matrix exhibits Toeplitz structure as explored in Araujo et al. (2021), where the singular values are computed. We can also mention the significant work of Singla and Feizi (2021a) regarding Lipschitz bounds of convolutions.

Reshaped Kernel Method (RKO)

This method is one of the first proposed, and also of of the simplest. It amounts to orthogonalize the kernel matrix. The order-4 tensor kernel of size $C \times M \times k \times l$ is reshaped into 2D matrix $Ckl \times M$ on which techniques from the previous section can be applied. The Lipschitz constant of the convolution is bounded by a constant factor of the spectral norm of its reshaped matrix (that depends on k, l, and of the padding) as specified in Cisse et al. (2017); Tsuzuku et al. (2018); Qian and Wegman (2018).

Cayley Transform

Trockman and Kolter (2021) proposed a method for learning orthogonal convolution based on Cayley transform in the Fourier domain. Thus even if the kernel size (# of weight) is $C^2.k^2$, the receptive field for each output is the full input image $C.H.W$. At inference, the convolution still requires computation in the Fourier domain. This method has the disadvantage of requiring forward and inverse FFT transforms. The algorithm is described in Alg. 3.

Algorithm 3 Convolutional Cayley transform

Require: $W \in \mathbb{R}^{M\times C\times k\times k}$, case $M = C$, $X \in \mathbb{R}^{C\times H\times W}$, $H = W$
Ensure: $Y \in \mathbb{R}^{M\times H\times W}$ output of an orthogonal convolution
$\hat{X}[i] = \text{FFT}(X[i]), i \in [0, C[$ ▷ $\hat{X} \in \mathbb{C}^{C\times H\times H}$
$\hat{W}[i,j] = \text{FFT}(Pad(W[i,j], (H,H)), (i,j) \in [0, C[^2$ ▷ $\hat{W} \in \mathbb{C}^{C\times C\times H\times H}$
$\hat{W}[i,j] = \text{Cayley}(\alpha[i,j]\hat{W}[i,j]/||\hat{W}[i,j]||)$ ▷ $\hat{W} \in \mathbb{C}^{C\times C\times H\times H}$
$\hat{Y}[i] = \hat{W}[i]\hat{X}$ ▷ $\hat{Y} \in \mathbb{C}^{C\times H\times H}$
$Y[i] = \text{FFT}^{-1}(\hat{Y}[i]), i \in [0, C[$ ▷ $Y \in \mathbb{R}^{C\times H\times H}$ **return** Y

Exponential map (SOC)

Singla and Feizi (2021b) introduced Skew Orthogonal Convolutions (SOC), based on the exponential map. From the original unconstrained kernel W, they first construct a convolution kernel W_s such that the corresponding Jacobian $\mathbf{J}_s$ is skew-symmetric. They compute $\Theta = \exp(\mathbf{J}_s)$, which is orthogonal.

1. They prove that the Jacobian $\mathbf{J}_s$ of a convolution is skew-symmetric if and only if the convolution kernel W_s is built from any kernel W as

$$\text{conv}_{W_s} = \text{conv}_W - \text{conv\_transpose}_W. \tag{2.24}$$

   where the conv_transpose operation is defined as

$$\text{conv\_transpose}(W)_{i,j,m,c} = W_{k-1-i,l-1-j,c,m},$$

   i.e., flipping the kernel along the horizontal and vertical directions, and transposing dimensions corresponding to input and output channels. This analogous to the method exposed in section 2.3.2. Note that this operation brings some constraints on W: we must have the same number of input and output channels ($M = C$) and an odd kernel size (k and l must be odd).
2. The (flattened) orthogonal convolution is expanded as

$$\Theta x = \exp(\mathbf{J}_s)\, x = \sum_{k=0}^{\infty} \frac{\mathbf{J}_s^k x}{k!} = x + \frac{W_s \star x}{1!} + \frac{W_s \star^2 x}{2!} + \frac{W_s \star^3 x}{3!} + \dots \tag{2.25}$$

   In practice, $\exp(\mathbf{J}_s)$ is not explicitly computed. Instead, an approximation of the product $\exp(\mathbf{J}_s)\, x$ is obtained with a truncated Taylor expansion. Successive convolution operations are applied: $W_s \star x$, $W_s \star^2 x = W_s \star (W_s \star x)$, etc. Since the parametrization is implicit, the successive convolutions must be done for any new x, in particular at inference time. The authors suggest to keep 6 terms in the power series during training (for speed) and 12 terms for inference (for precision). Moreover, spectral normalization is applied on W_s to ensure that the matrix falls within the convergence radius of the Taylor expansion.

Note that this formulation holds for square matrix Θ, especially for $M = C$ and $S = 1$. The authors in Singla and Feizi (2021b) propose solutions to handle the more general cases. Strides are handled using an invertible downsampling as pre-processing. If the convolution contains more output channels than input channels, i.e. $M > C$, the convolution kernel is built with M filters, and the input is zero-padded with $M - C$ channels. If the convolution contains fewer output channels than input channels, i.e. $M < C$, the convolution kernel is built with C filters, and the output is truncated to keep the first M channels. In Singla and Feizi (2022), an improvement is proposed to speed up the computation of the gradient: compute the gradient only from the first-order term.

LOT (Layer-wise orthogonal training) Xu et al. (2022b)

This method is a direct extension of Newton's iterations of section 2.3.2 to convolutions. As Cayley transform, it requires to work in Fourier space and suffers from the same drawbacks that it implies: a FFT and an inverse FFT are required.

2.4 Implementation

In this section we discuss some practical tricks involved in the implementation of Lipschitz networks.

2.4.1 Practical considerations

Residual connections are Lipschitz but prone to vanishing gradients.

Remark 2.6. Residual connections

If f verifies $\|\nabla_x f(x)\| = 1$ almost everywhere, and if g verifies $\|\nabla_x g(x)\| = 1$ almost everywhere, then $\|\nabla_x(\frac{1}{2}f(x) + \frac{1}{2}g(x))\| < 1$ in general, unless $\nabla_x f(x) = \nabla_x g(x)$. Taking $f(x) = x$ we end up with residual connections, for which ensuring $\|\nabla_x(\frac{1}{2}f(x) + \frac{1}{2}g(x))\| = 1$ almost everywhere is not possible unless $f = g$. Seemingly easy tasks like "copy and duplicate" such as $x \mapsto [x, x]$ (which are common for networks with parallel branches) **are not** GNP.

Remark 6 essentially shows that the set of GNP layers is not stable by sum or other common operations. This makes their practical implementations, and the demonstration of universal approximation theorems trickier.

Vanishing and Exploding gradients have been a long-time issue in the training of neural networks. The latter is usually avoided by regularizing the weights of the networks and using bounded losses, while the former can be avoided using residual connections (such ideas can found on LSTM Gers et al. (1999) or ResNet He et al. (2016)). On Gradient Norm Preserving (GNP) networks (orthogonal networks with GroupSort activation such as the ones of *Deel.lip* library), we can guarantee the absence of exploding gradient:

Proposition 1 (No exploding gradients Li et al. (2019a))**.** *Assume that $f = h^M \circ h^{M-1} \circ \ldots \circ h^2 \circ h^1$ is a feed-forward neural network and that each layer h^i is 1-Lipschitz, where h^i is either a 1-Lipschitz affine transformation $h^i(x) = W^i x + B^i$ either a 1-Lipschitz activation function. Let $\mathcal{L} : \mathbb{R}^k \times \mathcal{Y} \to \mathbb{R}$ the loss function. Let $\tilde{y} = f(x)$, $H^i = h^i \circ h^{i-1} \circ \ldots \circ h^2 \circ h^1$ and $H^0(x) = x$. Then we have:*

$$\|\nabla_{W^i}\mathcal{L}(\tilde{y}, y)\| \leq \|\nabla_{\tilde{y}}\mathcal{L}(\tilde{y}, y)\| \times \|H^{i-1}(x)\|, \tag{2.26}$$

$$\|\nabla_{B^i}\mathcal{L}(\tilde{y}, y)\| \leq \|\nabla_{\tilde{y}}\mathcal{L}(\tilde{y}, y)\|. \tag{2.27}$$

To prove **Proposition 1** we just need to write the chain rule.

Proof. The gradient is computed using chain rule. Let θ be any parameter of layer h^i. Let $h^j_\perp$ be a dummy variable corresponding to the input of layer h^j, which is also the output of layer h^{j-1}. Then we have:

$$\nabla_\theta \mathcal{L}(\tilde{y}, y) = \nabla_{\tilde{y}} \mathcal{L}(\tilde{y}, y) M (J_\theta h^j(H^{i-1}(x))). \tag{2.28}$$

with $M = \left(\prod_{j=M}^{i+1} J_{h^j_\perp} h^j(H^{j-1}(x))\right)$. As the layers of the neural network are all 1-Lipschitz, we have:

$$\|J_{h^j_\perp} h^j(H^{j-1}(x))\| \leq 1.$$

Hence we get the following inequality:

$$\|\nabla_\theta \mathcal{L}(\tilde{y}, y)\| \leq \|\nabla_{\tilde{y}} \mathcal{L}(\tilde{y}, y)\| \|J_\theta h^j(H^{i-1}(x))\|. \tag{2.29}$$

Finally, for $h^i(H^{i-1}(x)) = W^i H^{i-1}(x) + B^i$ we replace θ by the appropriate parameter which yields the desired result. □

There is still a risk of vanishing gradient, which strongly depends of the loss $\mathcal{L}$.

2.4.2 Frameworks in the wild

Deel-lip. Most experiments done in the thesis rely on the `deel-lip`[2] library (Serrurier et al., 2021) to enforce Lipschitz constraints in practice, with Reshaped Kernel Orthogonalization (RKO) for fast and near-orthogonal convolutions (Li et al., 2019b). Its design follows ideas of Anil et al. (2019). The networks use 1) orthogonal matrices and 2) GroupSort2 activation. Orthogonalization is enforced using Spectral normalization (Miyato et al., 2018) and Björck algorithm (Björck and Bowie, 1971).

Deel-lip relies on Power Iteration for fast computation of the spectral norm. The leading eigenvector is cached from one gradient step to another to speed up computations. Indeed, by continuity, the leading singular values of W_t and $W_{t+1} := W_t + \eta \nabla_W \mathcal{L}$ are very close when the step size η is small. This is an instance of amortized optimization (Amos et al., 2023).

- **Torch-lip** is the Pytorch implementation of deel-lip. It contains fewer features and lags a little behind the *Tensorflow* implementation.
- **Pytorch** standard API provides few building blocks for Lipschitz networks on tabular data, such as spectrally normalized matrices or matrices with orthogonal constraints, thanks to the `parametrization` API.
- **Other libraries.** At the time of writing (2023), to the best of my knowledge, no other library implement neural networks with Lipschitz constraints. Most projects related to Lipschitz networks parametrization give public code that companions a paper, but to date, no project gathers every implementation in one place with a common API and a comprehensive comparison.

In this chapter, we performed a brief review of the state of the art regarding the parametrization of Lipschitz networks. In the next chapter, we take a detour by the field of "optimization as a layer" and "implicit differentiation" before going back to Lipschitz networks. The reason is that these paradigms give tools to efficiently compute the projection of a matrix onto the Stiefel manifold and to back-propagate through this operation. We also explore other applications of these tools.

[2] `https://github.com/deel-ai/deel-lip` distributed under MIT License.

Chapter 3

Optimization as a layer

This chapter delves into a recent trend of deep learning, that can be summarized under the name of "optimization as a layer" and whose terminology can be attributed to Amos and Kolter (2017). The spirit of the idea is older, and be found in works like Genevay et al. (2018). Even the min-max formulations of GAN (Goodfellow et al., 2014) can be seen as an instance of this framework. The idea is to use the *solution to a constrained optimization problem* as a layer in a neural network, as follow:

$$\begin{aligned} y(\theta, x) \in \quad & \underset{y \in \mathbb{R}^d}{\arg\min}\, f(\theta, x, y) \\ \text{such that} \quad & g(\theta, x, y) \leq \mathbf{0} \\ & h(\theta, x, y) = \mathbf{0} \end{aligned} \tag{3.1}$$

where θ are parameters, x an input, $y(\theta, x)$ a function of input and parameters (like any layer), f the objective function, and g and h the constraints. Typically, the functions f, g, h are chosen such that $y(\theta, x)$ is uniquely defined for *almost all* values of x and θ, even though exceptions may exist.

Exemple 3.1. "Optimization layers"

This framework is versatile. For example:

- It can be used to design a loss function, in which case $y(\theta, x)$ is a scalar, and θ is just a hyper-parameter of the algorithm. For example, it is the case of Sinkhorn algorithm (Genevay et al., 2018).
- It can be used as a layer parametrized by θ. *Softmax* is an example of such function, as detailed in equation 4.31 in section 4.4. Even non-linearities like ReLU can me modelized as $\arg\min_{y \geq \mathbf{0}} \|y - x\|_2$.

Layers inherit the property and the structure of the optimization problem. This is a way to enforce constraints in the network, and to leverage all the work done in the field of optimization to design new layers. In either case, the application of the "deep learning framework" requires to compute the derivatives

$$\frac{\partial y}{\partial x} \quad \text{or} \quad \frac{\partial y}{\partial \theta}$$

to allow for optimization with gradient descent.

The first strategy, called *unrolling*, consists in using an Algorithm $\mathcal{A}$ to compute $y(\theta, x) := y_T$.

An algorithm can be seen as a discrete dynamical system acting on a state y_t:

$$\begin{cases} y_0 \text{ arbitrary} \\ y_{t+1} := \mathcal{S}(\theta, x, y_t) \end{cases} \tag{3.2}$$

where $\mathcal{S}$ is the "step" function, that carries one iteration of computation. If $\mathcal{S}$ is differentiable, then the Autodiff can be used to back-propagate through algorithm iterations and will return a "derivative". Here, things start to get ugly because one must ensure that the derivative computed this way is indeed the derivative of the original's problem. Indeed, in the typical situation $\mathcal{S}$ is differentiable *almost* everywhere, and y_T is not the true optimum but rather an approximation of thereof (Bolte et al., 2022). These considerations are outside the scope of this work but illustrate the difficulty related to this paradigm. An alternative method to compute derivatives, named *implicit differentiation*, is exposed in the next section.

This chapter is structured as follow:

1. **First**, we expose the implicit function theorem, and its practical implementation in Jaxopt library with AutoDiff.
2. **Then**, we showcase an application of implicit differentiation to Non-Negative Matrix Factorization, using ADMM, with application to explainability of neural networks.
3. **Finally**, we show how to accelerate the computations of the derivatives of the projection onto Stiefel manifold using implicit differentiation.

In Appendix A.1 we also showcase an example of the *manual differentiation* of differentiable images data-augmentations.

Contents

3.1 Differentiation of optimization problems

Implicit differentiation is a technique based on the *implicit function theorem*, that can be used to compute the derivative of functions *implicitly* defined as the root, the fixed point, or the solution to an optimization problem (Krantz and Parks, 2002; Griewank and Walther, 2008; Bell and Burke, 2008). In this section, we explain the technique and discuss its practical implementation in the *Jaxopt* library.

3.1.1 Implicit differentiation

The function $y(\theta, x)$ is *implicitly* defined. Under suitable assumptions on the optimization problem given in equation 3.1, the optimum $y(\theta, x)$ is the root of the so-called "optimality function" F. In the neighborhood of the root of F, and under suitable smoothness assumptions on F, the derivatives

of $y(\theta, x)$ can be computed using the derivatives of F. Implicit functions theorems are a family of results that relate these two quantities, with more or less restrictive hypotheses on F (Krantz and Parks, 2002). We detail below the most frequent version, that is of interest in the context of deep learning.

Theorem 1 (Implicit Function Theorem)**.** *Let $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ be a continuously differentiable function, and suppose that there exist points $(a, b) \in \mathbb{R}^n \times \mathbb{R}^m$ such that $F(a, b) = 0$. We note:*

- *$\partial_1 F(a, b) \in \mathbb{R}^{m \times n}$ be the Jacobian matrix of F w.r.t the first variable at the point (a, b),*
- *$\partial_2 F(a, b) \in \mathbb{R}^{m \times m}$ be the Jacobian matrix of F w.r.t the second variable at the point (a, b),*

and we assume that $\partial_2 F(a, b)$ is invertible. Then, there exist open neighborhoods U of a in $\mathbb{R}^n$, and a continuously differentiable function $f : U \to \mathbb{R}^m$ such that for all $x \in U$, we have $F(x, f(x)) = 0$ and $f(a) = b$. Moreover, the Jacobian $\partial f(a) \in \mathbb{R}m \times n$ verifies:

$$\partial f(a) = -\left(\partial_2 F(a, b)\right)^{-1} \partial_1 F(a, b). \tag{3.3}$$

We consider an optimization problem with optimum $x(\theta)$, objective function $f(\cdot, \theta)$, equality constraints $g(\cdot, \theta)$ and inequality constraints $h(\cdot, \theta)$:

$$\begin{aligned} x(\theta) \in \quad & \underset{x \in \mathbb{R}^n}{\arg\min}\, f(x, \theta) \\ \text{such that} \quad & g(x, \theta) \leq \mathbf{0} \\ & h(x, \theta) = \mathbf{0} \end{aligned} \tag{3.4}$$

The optimality function F is typically obtained with the Karush–Kuhn–Tucker (KKT) conditions Karush (1939); Kuhn and Tucker (1951). The KKT conditions apply in various situations, depending on the hypothesis made on f, g, and h. For example, for convex optimization problems, Slater's condition is one of them (Slater, 2013): this requires that f and g are convex functions and that h is affine. In every case, the optimality function F takes the following form:

$$F((x, \lambda, \mu)), \theta) = \begin{cases} \nabla_x f(x, \theta) + \lambda^T \partial_1 g(x, \theta) + \mu^T \partial_1 h(x, \theta), & \text{stationarity} \\ h(x, \theta), & \text{primal feasability} \\ \lambda \circ g(x, \theta), & \text{complementary slackness} \end{cases} \tag{3.5}$$

where λ and μ are the dual variables (Lagrange multipliers) of the problem. One can check that the triplet $(x(\theta), \lambda(\theta), \mu(\theta))$ is indeed a root of $(x, \lambda, \mu) \mapsto F((x, \lambda, \mu)), \theta)$. Note that the optimality function F requires the dual variables, not only the primal ones. Other simpler example for F is in fixed point finding: if x^* is a fixed point of G, i.e. $G(x^*) = x^*$, then x^* is indeed the root of $F(x) = G(x) - x$, and reciprocally. The details regarding these formulations can be found in Blondel et al. (2022).

Implicit differentiation in literature. Implicit differentiation has gathered considerable attention in recent years, with applications ranging from hyper-parameter optimization (Lorraine et al., 2020), implicit layers (Bai et al., 2019; El Ghaoui et al., 2021; Fung et al., 2022), optimization as a layer (Amos and Kolter, 2017; Niculae and Blondel, 2017) for structured prediction. Implicit differentiation has been used for a long time, and was historically named "*Adjoint state method*" in other fields, notably optimization (Céa, 1986).

3.1.2 Jaxopt library

Remark 3.1. What's new in deep learning?

As mentioned previously, the implicit function theorem has been used for a long time in the field of optimization. In deep learning, the novelty is that AutoDiff allows for computation of the Jacobians $\partial_2 F(a,b)$ and $\partial_1 F(a,b)$ automatically, instead of manually. Therefore, not implicit differentiation benefit *to* deep learning by offering ways to train implicit layers or hyper-parameters, but it also benefits *from* deep learning AutoDiff. This offers the possibility to use deep learning frameworks (like *Tensorflow*, *Pytorch* or *Jax*) for optimization tasks, outside the context of *learning* (which involves more than just optimization).

Jaxopt (Blondel et al., 2022) allows efficient computation of $\frac{\partial x}{\partial \theta} = -(\partial_1 F)^{-1}\partial_2 F$. The matrix $(\partial_1 F)^{-1}$ is never explicitly computed – that would be too costly. Instead, the system

$$(\partial_1 F)\frac{\partial x}{\partial \theta} = -\partial_2 F \tag{3.6}$$

is solved with *indirect* linear system solvers. Among indirect linear solvers we can mention conjugate gradient (Hestenes and Stiefel, 1952), generalized minimal residual method (Saad and Schultz, 1986) or biconjugate gradient stabilized method (Van der Vorst, 1992). They are called "indirect" because they only require access to the linear operator $x \mapsto Mx$ but not . We propose to rely on Jacobian Vector Products (JVP) $v \mapsto (\partial_1 F)v$. These JVP are efficiently computed by Autodiff. Often, we are not interested in the Jacobian $\frac{\partial x}{\partial \theta}$ directly, but rather a gradient

$$\nabla_\theta \mathcal{L} = (\nabla_x \mathcal{L})\frac{\partial x}{\partial \theta}. \tag{3.7}$$

of some loss function $\mathcal{L}$. Therefore, the *matrix* linear system (where the unknown is a Jacobian) becomes a *vector* linear system where the unknown is a cotangent vector.

In the Jaxopt framework, the user only has to implement f, g, and h in Jax, and the library takes care of the rest: AutoDiff handles everything transparently. This recipe allows effortless computation of the derivative of dozens of optimization tasks. The "optimization layers" can be seamlessly incorporated in computation graphs and be compatible with backpropagation. Moreover, they can run on GPU thanks to Jax. Finally, computations can be parallelized transparently with 'jax.vmap' transformation, which allows solving dozens of optimization problems in parallel.

My contributions to this framework are highlighted below[1].

Fixed point computations and Anderson acceleration

For a contractive mapping T, i.e. $T : \mathbb{R}^n \to \mathbb{R}^n$ fulfill $\|T(x) - T(y)\|_2 < \|x - y\|_2$, the fixed point x^* exist and is unique: $T(x^*) = x^*$. This is the Banach fixed point theorem (Banach, 1922). The proof is constructive and relies on the sequence $x_{n+1} := T(x_n)$, sometimes referred to as "Picard iterations" in the context of ODE solving (Coddington et al., 1956). Anderson acceleration with history size m is an algorithm that performs a *weighted* average of the previous iterates $x_t, x_{t+1}, \dots x_{t+m}$ to interpolate the next one x_{t+m+1}, instead of taking the last in the sequence. We illustrate this on ODE solving task in Figure 3.1. This algorithm can also be used to speed up the convergence of fixed-point solvers under some circumstances. Indeed, the optimum of an iterative solver is also its fixed point. We illustrate this on the "Block coordinate descent" algorithm, following Bertrand and Massias (2021), in Figure 3.2. Finally, we can mention *Deep Equilibrium Models* (DEQ) (Bai

[1] It can also be found on `https://jaxopt.github.io/stable/changelog.html`

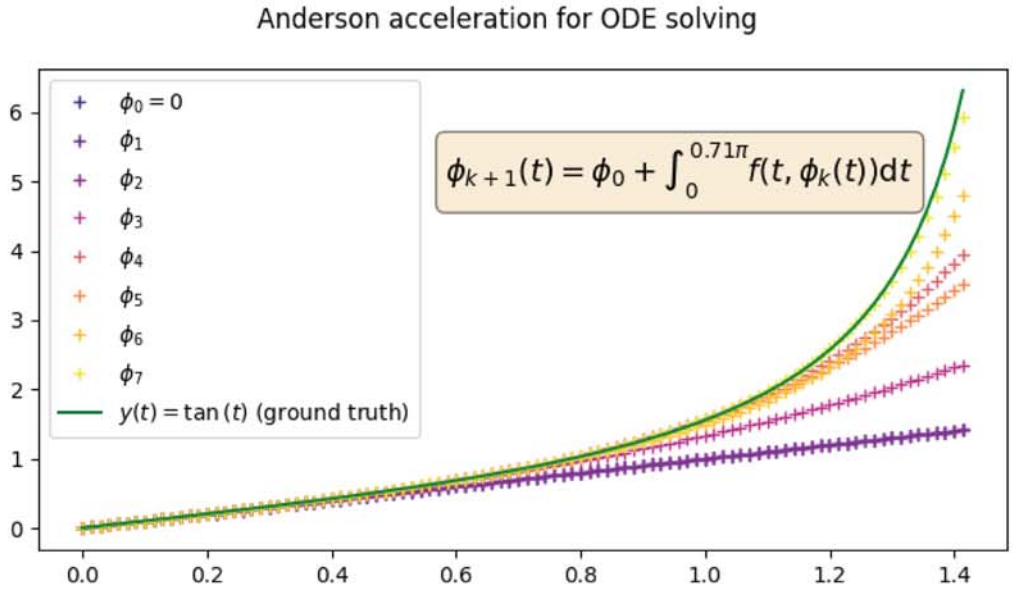


Figure 3.1: **ODE solving with Picard method**. According to Picard–Lindelof theorem (Coddington et al., 1956), the solutions of ODE $y'(t) = f(t, y(t))$ with $y(t_0) = y_0$ are the fixed point of the operator $\mathcal{T}(y)(t) = y_0 + \int_{t_0}^{t} f(s, y(s))ds$. By discretizing the interval $[t_0, t]$ into n bins, the solution takes the form $y \in \mathbb{R}^n$, and the operator takes the form $\mathcal{T} : \mathbb{R}^n \to \mathbb{R}^n$. We accelerate the fixed-point solving $\phi_{k+1} = T(\phi_k)$ with Anderson extrapolation. This plot can also be found in *Jaxopt* example gallery.

et al., 2019) among applications of Fixed point iterations solvers. In a DEQ, the layer is defined as $z^*_\theta(x)$ where $z^*_\theta(x)$ is the fixed point of the function $z \mapsto T_\theta(z, x)$, starting from $z_0 = x$. These models can be trained with implicit differentiation, using optimality function $F(z) = T_\theta(z, x) - z$. An implementation can be found in Jaxopt.

Stochastic optimization with Armijo line-search.

The step size η is a crucial hyper-parameter on every gradient-based algorithm. If it is too big, the training may diverge catastrophically. If it is too small, the algorithm may never reach the optimum. For example, Robbins and Monro (1951) consider a sequence η_t of decreasing step sizes such that $\sum_t \eta_t = +\infty$ and $\sum_t \eta_t^2 < +\infty$. Another strategy is to use a *linesearch*: at each step, the optimal step size η_t is chosen by optimizing a certain criterion. Armijo line search stands out (Armijo, 1966). It has been adapted to the stochastic setting in Vaswani et al. (2019). This search looks for η_t such that $f(\theta_t - \eta_t \nabla_\theta f(\theta_t)) \leq f(\theta_t) - c\eta_t \|\nabla_\theta f(\theta_t)\|_2^2$ with $c > 0$ an hyper-parameter. Polyak step sizes (Polyak, 1964; Loizou et al., 2021) are much simpler and use the rule $\eta_t \propto \frac{f(\theta_t) - f^*}{\|\nabla_x f\|^2}$. Note that the step sizes given in Equation 5.11 in Section 5.2.2 can be interpreted as deterministic Polyak step sizes. The comparison between the different algorithms is given on Fashion-Mnist in Figure 3.3.

Quadratic programming.

The last solver implemented during my internship was the celebrated Operator Splitting Quadratic Program (OSQP) solver presented in Stellato et al. (2020); Banjac et al. (2019). This is a solver for

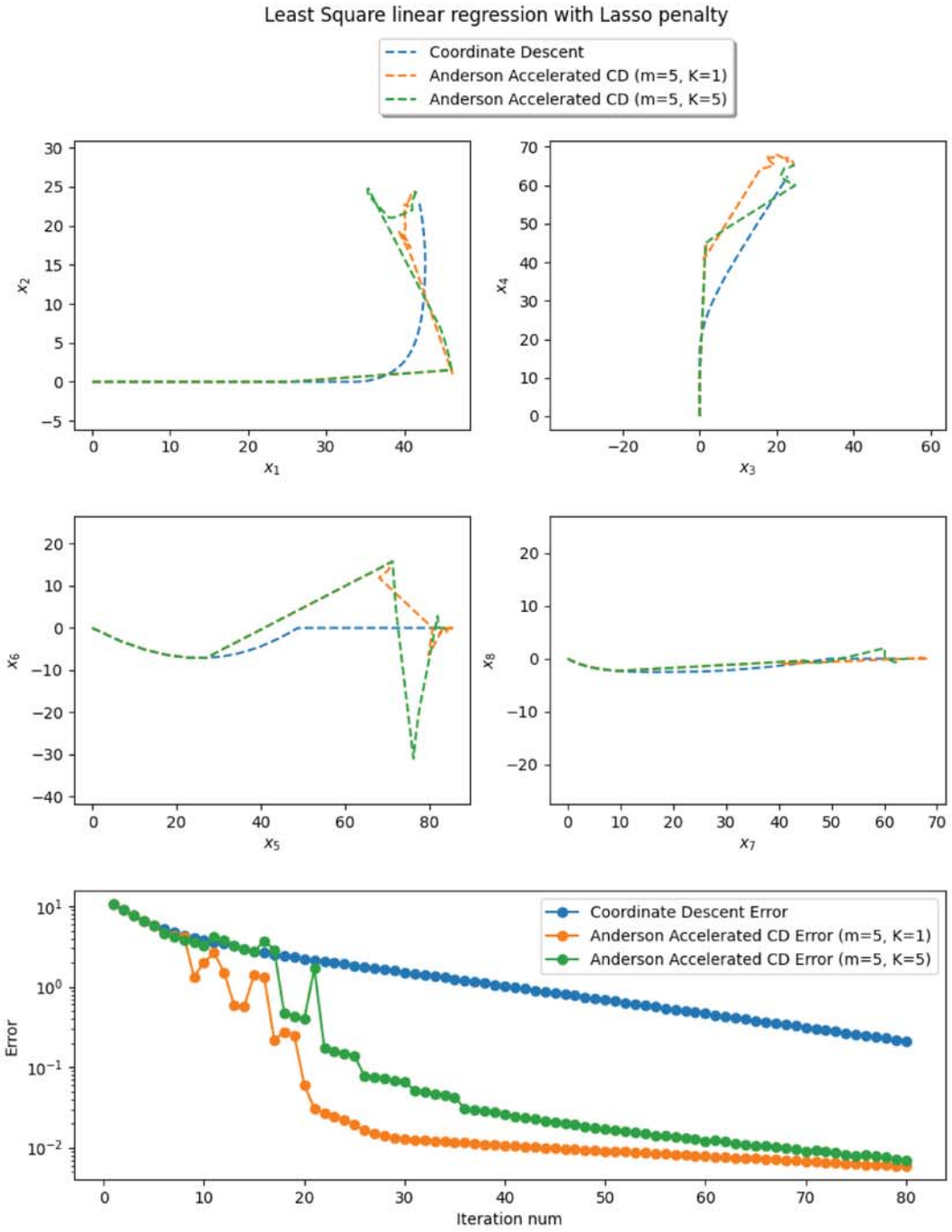


Figure 3.2: **Anderson acceleration of Block Coordinate Descent algorithm**. It follows the setting of Bertrand and Massias (2021). m denotes the history size, while K denotes the *mixing frequency*, i.e. the frequency at which the extrapolation weights are updated. This plot can be found in *Jaxopt* library gallery.

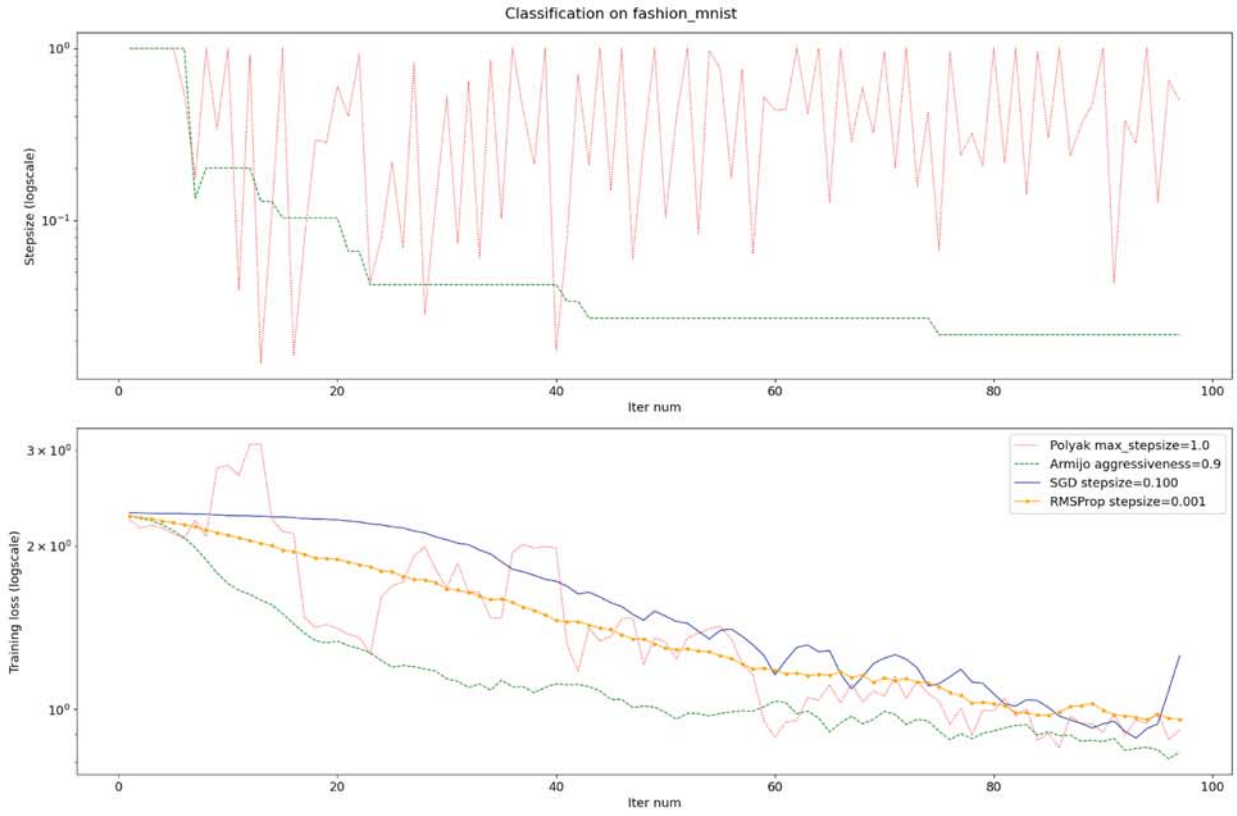


Figure 3.3: **Comparison of SGD solvers on Fashion-Mnist with a simple CNN.**

convex quadratic programs, i.e. optimization problems of the form:

$$\begin{aligned} &\arg\min_{x\in\mathbb{R}^d} f(x) := \frac{1}{2}x^T Q x + c^T x \\ &\text{such that } l \leq Ax \leq u \end{aligned} \tag{3.8}$$

with $Q \in \mathbb{R}^{d\times d}$ a Positive Definite Matrix (which guarantees that the problem is convex), with $c \in \mathbb{R}^d$ an arbitrary vector, with $A \in \mathbb{R}^{n\times d}$ the matrix of n linear constraints, with lower bounds $l \in \mathbb{R}^n$ and upper bounds $u \in \mathbb{R}^n$. This formulation is extremely general since one-sided inequalities constraints can be achieved with $l_i = -\infty$ and $u_i = +\infty$. It was re-implemented integrally in Jax with support for GPU, following guidelines of Schubiger et al. (2020), and with support for AutoDiff. Furthermore, the matrix representation of Q and c are not required: any function $f(x)$ promised to be quadratic convex (even with constant term) *and* differentiable can be used. Indeed, the vector c is the offset $c = \nabla_x f(\mathbf{0})$. And the operator $x \mapsto Qx$ is the function $x \mapsto \nabla_x(f(x) - c^T x)$. When Q is sparse this translates into enormous gains since the coefficients Q_i are never directly needed. This algorithm made possible the re-implementation of an SVM solver from scratch, which can be found in Jaxopt's examples gallery.

3.2 Trivialization of the Stiefel Manifold

In this section, we explore the implicit differentiation of the projection onto the Stiefel manifold. The content of this section is unpublished. This echoes discussions of section 2.3. For simplicity, we focus on orthogonal matrices, i.e. square matrices. We are looking for the operator $\Pi : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ that solves the projection on the Stiefel manifold (in the *least Frobenius norm square* sense):

$$\begin{aligned} \Pi(W) = \arg\min_{\Theta\in\mathbb{R}^{n^2}} &\frac{1}{2}\|W - \Theta\|_F^2 \\ \text{such that } &\Theta^T\Theta - I = \mathbf{0}. \end{aligned} \tag{3.9}$$

This optimization problem is given by objective $f_W(\Theta) = \frac{1}{2}\|\Theta - W\|_F^2$ and constraint $g(\Theta) = \Theta^T\Theta - I$. Any projection algorithm can be used with implicit differentiation, including Björck algorithm (Björck and Bowie, 1971), see the section 2.3 for suggestions. We consider the problem defined in Equation (3.9). For clarity remember that $f : \mathbb{R}^{n^2} \to \mathbb{R}$ and $g : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$. The Lagrangian reads (with $\lambda \in \mathbb{R}^{n^2}$):

$$\mathcal{L}(\Theta) = f_W(\Theta) + \langle \Lambda, g(\Theta)\rangle_F. \tag{3.10}$$

Here $\langle \Lambda, g(\Theta)\rangle_F = \text{Trace}(\Lambda^T g(\Theta))$ is the Frobenius inner product. We can compute its derivative to obtain the *stationarity* conditions:

$$\nabla_\Theta \mathcal{L}(\Theta) = (\Theta - W) + \Theta(\Lambda + \Lambda^T) = 0. \tag{3.11}$$

This implies that at the optimum (with $\Theta^{-1} = \Theta^T$) we have:

$$\Theta(\Lambda + \Lambda^T) = W - \Theta \implies \text{sym}(\Lambda) = \frac{1}{2}(\Theta^T W - I) \tag{3.12}$$

The matrix $\text{sym}(\Lambda)$ can be parametrized with only $\frac{n(n+1)}{2}$ parameters since it is symmetrized with operation *sym*. We name this reduce set of parameters $\lambda \in \mathbb{R}^{n(n+1)/2}$ such that $\Lambda = \text{tri}(\lambda)$ with *tri* a linear operation that reshapes the flat vector λ into symmetric matrix.

Once Θ has been computed, for example with Björck algorithm, λ is recovered by extracting the diagonal part of $\frac{1}{2}(\Theta^T W - I)$. We can readily see that if $W = \Theta$ (i.e the matrix is already orthogonal) then $\lambda = 0$: there is no cost induced by the projection. Other cases are more interesting. This is compliant with the results of Dalmau-Cedeno and Oviedo (2017), up to a factor 2 on Λ (attributed to the leading $\frac{1}{2}$ in objective). Then we plug these expressions into the implicit differentiation framework. The implicit root function is defined as:

$$F((\Theta, \lambda), W) = \begin{cases} (\Theta - W) + 2\Theta \text{tri}(\lambda) = \mathbf{0}, & \text{stationnarity} \\ \Theta^T \Theta - I = \mathbf{0}, & \text{primal feasability.} \end{cases} \tag{3.13}$$

Here the quantity of interest is $\nabla_W \mathcal{L} = \frac{\partial(\Theta,\Lambda)}{\partial W} \nabla_{(\Theta,\Lambda)} \mathcal{L}$. We define:

$$x := \frac{\partial \Theta}{\partial W} \text{ and } y := \frac{\partial \Lambda}{\partial W} \tag{3.14}$$

that are order-4 tensors. The implicit function theorem yields the following:

$$[x, y] \frac{\partial F}{\partial(\Theta, \Lambda)} = -\frac{\partial F}{\partial W}. \tag{3.15}$$

Remark 3.2. Breaking down every term.

The term $\frac{\partial F}{\partial W}$ takes the simple form $\frac{\partial F}{\partial W} = (-I, \mathbf{0})$. The term $\frac{\partial F}{\partial(\Theta,\Lambda)}$ can be splitted in tuples $\frac{\partial F}{\partial \Theta}$ and $\frac{\partial F}{\partial \Lambda}$. They involve order-4 tensors which can be tricky to write down. Hence we rely on Vector Jacobian Product (VJP) to explicit their form, where $\frac{\partial F}{\partial(\Theta,\Lambda)}$ is seen as a linear operator that operates on its co-tangent vector $[x, y]$:

$$\begin{aligned} [x, y] \frac{\partial F}{\partial \Theta} &= x + x(\Lambda + \Lambda^T) + \Theta(y + y^T), \\ [x, y] \frac{\partial F}{\partial \Lambda} &= \Theta^T x + (\Theta^T x)^T. \end{aligned} \tag{3.16}$$

Note that equation 3.15 can be rewritten to make the $\nabla_W \mathcal{L}$ term appear:

$$\begin{aligned} \nabla_W \mathcal{L} \frac{\partial F}{\partial(\Theta, \Lambda)} &= -(\nabla_{(\Theta,\Lambda)} \mathcal{L}) \frac{\partial F}{\partial W} \\ \implies (\nabla_{(\Theta,\Lambda)} \mathcal{L}) \frac{\partial(\Theta, \Lambda)}{\partial W} \frac{\partial F}{\partial(\Theta, \Lambda)} &= -(\nabla_{(\Theta,\Lambda)} \mathcal{L}) \frac{\partial F}{\partial W}. \end{aligned} \tag{3.17}$$

We can manually expand this equation and obtain a closed form solution for x and y. Notice that $\nabla_\Lambda \mathcal{L} = \mathbf{0}$ for all practical applications, since the dual variables are often of little use. We note $u := \nabla_\Theta \mathcal{L}$ the cotangent vector of the projection. Let B and C be:

$$B = W^T \Theta = B^T, \tag{3.18}$$

$$C = \Theta^T u - u^T \Theta = 2\text{skew}(\Theta^T u). \tag{3.19}$$

We denote the anti-commutator matrix operator as $\{B, X\} := BX + XB$ which is a linear operator. Let A be the solution to the linear system:

$$A := \text{solve}_X(\{B, X\} = C).$$

Observe that A is a skew-symmetric matrix: it lies in a space of dimension $\frac{n(n-1)}{2}$ which is compliant with the dimension of the Stiefel manifold. Then we have:

$$\frac{\partial \mathcal{L}}{\partial W} = \Theta A.$$

The vector u is obtained with classical back-propagation in the network. The matrix product ΘA is inexpensive, as well as forming B and C. The main difficulty is solving the anti-commutator problem $\{B, X\} = C$. The linear operator $X \mapsto \{B, X\}$ is an order-4 tensor of size $n^2 \times n^2$ which makes its materialization in memory impracticable even for small values of n. This draws out methods based on direct inversion or factorization (e.g. Cholesky). Indirect methods such as conjugate gradient can be applied out of the box.

Remark 3.3. Analytical solution.

It is also possible to derive manually the solution to the problem using the work of Rabkin (2015) and (https://math.stackexchange.com/users/67071/robert lewis). Let $W = U\Sigma V^T$ be the Singular Value Decomposition (SVD) of W with $U^T U = I$, $V^T V = I$ and Σ the diagonal matrix of singular values. Then it is well known that $\Theta = \Pi(W) = \Pi(U\Sigma V^T) = UV^T$ is the orthogonal projection of W onto Stiefel manifold. Consequently, we have $B = V\Sigma V^T$. This allows to rewrite A as function of singular values $\sigma_i \in \Sigma$:

$$\tilde{C} = V^T C V, \qquad \tilde{A}_{ij} = \frac{\tilde{C}_{ij}}{\sigma_i + \sigma_j}, \qquad A = V \tilde{A} V^T. \tag{3.20}$$

The solution is guaranteed as long as $\sigma_i + \sigma_j \neq 0$ for all pairs i, j. In particular, the null space of W should be of dimension at most 1. Since $\mathrm{GL}_n(\mathbb{R})$ is dense in $M_n(\mathbb{R})$ this can be guaranteed numerically by adding a random perturbation $W + \mathcal{E}$ of small norm $\|\mathcal{E}\| \ll 1$.

It is possible to compute either the eigenvalue decomposition of $B = W^T\Theta = V\Sigma V^T$, either its Schur decomposition. This is cheaper than computing the SVD of W. Indeed the factor U is not required. And since B is a symmetric matrix, more efficient factorization and decomposition algorithms are available. Note that the SVD of W allows to compute Θ efficiently. However preliminary tests showed that it was not worth the overhead during forward pass and that Björck projection should be preferred in every case.

Exemple 3.2. Speed of the implicit differentiation.

We perform a benchmark by computing random Vector Jacobian Products $\nabla_W \mathcal{L} = \frac{\partial(\Theta,\Lambda)}{\partial W} \nabla_{(\Theta,\Lambda)} \mathcal{L}$ on GPU. The results of the benchmark are summarized in figure 3.4. We notice that implicit differentiation with conjugate gradient is particularly inefficient. However implicit differentiation with eigenvalue decomposition matches the performance of unrolled Björck iterations for medium-size matrices ($n < 1000$) and outperform it for large matrices. The gap is even more significant for matrices of spectral norm greater than one: the overhead induced by power iteration to estimate the highest eigenvalue is non-negligible. The unrolling of Power Iteration and Björck algorithm leads to Out Of Memory (OOM) error on GPU when $n > 2000$ (see figure 3.5), with Colab default's hardware. This is because the number of iterations required to converge is huge.

Example 2 shows that implicit differentiation is only advantageous for enormous matrices (extremely wide neural networks), which severely limits the appeal of the method. The implementation

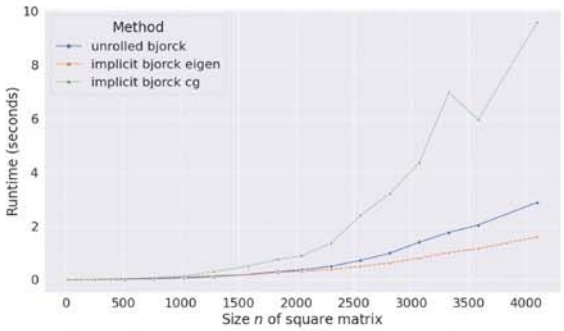


Figure 3.4: **Wall clock time** cost of computing a VJP $\nabla_W \mathcal{L} = \frac{\partial(\Theta,\Lambda)}{\partial W} \nabla_{(\Theta,\Lambda)} \mathcal{L}$ with three different methods, **on random matrices with unitary norm**. "unrolled" refers to the back-propagation through Bjorck iterations. "eigen" relies on the eigenvalue decomposition of the symmetric matrix B. "cg" solves the anti-commutator matrix equation with conjugate gradient.

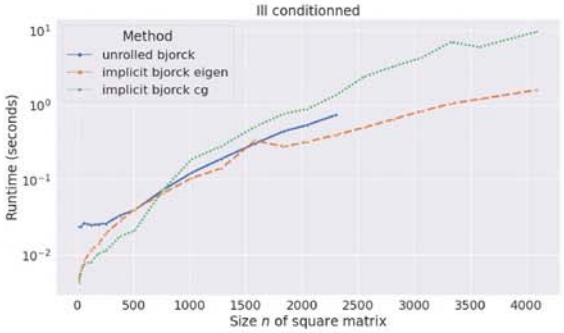


Figure 3.5: **Wall clock time** cost of computing a VJP $\nabla_W \mathcal{L} = \frac{\partial(\Theta,\Lambda)}{\partial W} \nabla_{(\Theta,\Lambda)} \mathcal{L}$ with three different methods on random matrices, with **ill-conditioned spectrum**. The line is interrupted when an out-of-memory error is raised. The wallclock runtime is typically higher for ill-conditioned matrices since solving the linear system takes more time. Similarly, Björck projection takes more time to converge.

is sketched below.

```
@tf.custom_gradient
def implicit_stiefel(W, u, adjustment_coef):
  theta = Pi(W, u, adjustment_coef)  # overhead here.
  theta = tf.stop_gradient(theta)

  def dPi_dW(dL_dTheta):
    B = tf.transpose(W) @ theta
    OTu = tf.transpose(theta) @ dL_dTheta
    C = OTu - tf.transpose(OTu)
    Sigma, V = tf.linalg.eigh(B)  # overhead here.
    V = V * tf.sign(Sigma)
    Sigma = tf.abs(Sigma)
    VT = tf.transpose(V)
    lbdas = tf.expand_dims(Sigma, axis=0) + tf.expand_dims(Sigma, axis=-1)
    epsilon = 1e-9
    lbdas = tf.where(tf.abs(lbdas) <= epsilon, epsilon, lbdas)
    C_tilde = VT @ C @ V
    A_tilde = C_tilde / lbdas
    A = V @ A_tilde @ VT
    dL_dW = theta @ A
    return dL_dW

  return theta, dPi_dW
```

The experiments of figures 3.4 and 3.5 show that the method is only interesting in the very large scale regime, which does not frequently happen in practice. For neural networks this big, it might be more reasonable to use the SDP-based *direct parametrization* discussed in section 2.2.4.

3.3 Differentiable Non Negative Matrix Factorization

In this section, we show how to apply implicit differentiation to an NMF solver based on ADMM. We conclude by showcasing an application of this algorithm to the field of XAI.

Non Negative Matrix (Lee and Seung, 1999) factorization decomposes the positive features vector $\mathbf{A} \in \mathbb{R}^{n \times p}$ of n examples lying in dimension p, into a product of positive low rank matrices $\mathbf{U}(\mathbf{A}) \in \mathbb{R}^{n \times r}$ and $\mathbf{W}(\mathbf{A}) \in \mathbb{R}^{p \times r}$ (with $r << \min(n, p)$), i.e the solution to the problem:

$$\min_{\mathbf{U} \geq 0, \mathbf{W} \geq 0} \frac{1}{2} \|\mathbf{A} - \mathbf{U}\mathbf{W}^T\|_F^2. \tag{3.21}$$

In this section, we explore how to **(i)** solve NMF with ADMM (Boyd et al., 2011) **(ii)** compute the derivatives with implicit differentiation of the ADMM solution, based on all the KKT variables.

Warning 3.1. Coordinate descent.

Our approach is not the only way to solve the NMF problem. Coordinate descent can also be used for NMF, as in scikit-learn's default implementation (Cichocki and Phan, 2009). Moreover, this alternative formulation also opens a path for implicit differentiation, not based on KKT conditions, but rather on the fixed point of a proximal operator. The benchmark between the two methods remains to be done.

3.3.1 Alternating Direction Method of Multipliers

For simplicity, we used a non-regularized version of the NMF objective, following Algorithms 1 and 3 in Huang et al. (2016b), based on ADMM. ADMM framework transforms the non-linear equality constraints into indicator functions $\boldsymbol{\delta}$. Auxiliary variables $\tilde{\mathbf{U}}, \tilde{\mathbf{W}}$ are also introduced to separate the optimization of the objective on the one side, and the satisfaction of the constraint on $\mathbf{U}, \mathbf{W}$ on the other side. The equality constraints $\tilde{\mathbf{U}} = \mathbf{U}, \tilde{\mathbf{W}} = \mathbf{W}$ are linear and easily handled by the ADMM framework through the associated dual variables $\bar{\mathbf{U}}, \bar{\mathbf{W}}$. In our case, the problem in Equation 3.21 is transformed into:

$$\begin{aligned} &\min_{\mathbf{U}, \tilde{\mathbf{U}}, \mathbf{W}, \tilde{\mathbf{W}}} \frac{1}{2} \|\mathbf{A} - \tilde{\mathbf{U}}\tilde{\mathbf{W}}^T\|_F^2 + \boldsymbol{\delta}(\mathbf{U}) + \boldsymbol{\delta}(\mathbf{W}), \\ &\quad s.t. \;\; \tilde{\mathbf{U}} = \mathbf{U}, \tilde{\mathbf{W}} = \mathbf{W} \\ &\text{with } \boldsymbol{\delta}(\boldsymbol{H}) = \begin{cases} 0 \text{ if } \boldsymbol{H} \geq 0, \\ +\infty \text{ otherwise.} \end{cases} \end{aligned} \tag{3.22}$$

Remark 3.4. The "useless" variables of ADMM framework.

Note that $\tilde{\mathbf{U}}$ and $\mathbf{U}$ (resp. $\tilde{\mathbf{W}}$ and $\mathbf{W}$) seem redundant: they are meant to be equal thanks to constraints $\tilde{\mathbf{U}} = \mathbf{U}, \tilde{\mathbf{W}} = \mathbf{W}$. This is standard practice within ADMM framework: introducing redundancies allows to disentangling the (unconstrained) optimization of the objective on one side (with $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{W}}$), and constraint satisfaction on the other side with $\mathbf{U}$ and $\mathbf{W}$. During the optimization process the variables $\tilde{\mathbf{U}}, \mathbf{U}$ (resp. $\tilde{\mathbf{W}}, \mathbf{W}$) are different and only become equal in the limit at convergence. The dual variables $\bar{\mathbf{U}}, \bar{\mathbf{W}}$ control the balance between optimization of the objective $\frac{1}{2}\|\mathbf{A} - \tilde{\mathbf{U}}\tilde{\mathbf{W}}^T\|_F^2$ and constraint satisfaction $\tilde{\mathbf{U}} = \mathbf{U}, \tilde{\mathbf{W}} = \mathbf{W}$. The constraints are simplified at the cost of a non-smooth (and even a non-finite) objective

function $\frac{1}{2}\|\mathbf{A} - \bar{\mathbf{U}}\bar{\mathbf{W}}^T\|_F^2 + \boldsymbol{\delta}(\mathbf{U}) + \boldsymbol{\delta}(\mathbf{W})$ due to the term $\boldsymbol{\delta}(\mathbf{U}) + \boldsymbol{\delta}(\mathbf{W})$.

ADMM proceeds to create a so-called *augmented Lagrangian* with l_2 regularization $\rho > 0$:

$$\begin{aligned}\mathcal{L}(\mathbf{A}, \mathbf{U}, \mathbf{W}, \tilde{\mathbf{U}}, \tilde{\mathbf{W}}, \bar{\mathbf{U}}, \bar{\mathbf{W}}) =& \frac{1}{2}\|\mathbf{A} - \tilde{\mathbf{U}}\tilde{\mathbf{W}}^T\|_F^2 + \boldsymbol{\delta}(\mathbf{U}) + \boldsymbol{\delta}(\mathbf{W}) \\ &+ \bar{\mathbf{U}}^T(\tilde{\mathbf{U}} - \mathbf{U}) + \bar{\mathbf{W}}^T(\tilde{\mathbf{W}} - \mathbf{W}) \\ &+ \frac{\rho}{2}\left(\|\tilde{\mathbf{U}} - \mathbf{U}\|_2^2 + \|\tilde{\mathbf{W}} - \mathbf{W}\|_2^2\right).\end{aligned} \tag{3.23}$$

This regularization ensures that the dual problem is well posed and that it remain convex, even with the non smooth and infinite terms $\boldsymbol{\delta}(\mathbf{U}) + \boldsymbol{\delta}(\mathbf{W})$. Once again, this is standard practice within ADMM framework. The (regularized) problem associated to this Lagrangian is decomposed into a sequence of convex problems that alternate minimization over the $\mathbf{U}, \tilde{\mathbf{U}}, \bar{\mathbf{U}}$ and the $\mathbf{W}, \tilde{\mathbf{W}}, \bar{\mathbf{W}}$ triplets.

$$\mathbf{U}_{t+1} = \underset{\mathbf{U}=\tilde{\mathbf{U}}}{\arg\min} \frac{1}{2}\|\mathbf{A} - \tilde{\mathbf{U}}\mathbf{W}_t^T\|_F^2 + \boldsymbol{\delta}(\mathbf{U}) + \frac{\rho}{2}\|\tilde{\mathbf{U}} - \mathbf{U}\|_2^2. \tag{3.24}$$

$$\mathbf{W}_{t+1} = \underset{\mathbf{W}=\tilde{\mathbf{W}}}{\arg\min} \frac{1}{2}\|\mathbf{A} - \mathbf{U}_t\tilde{\mathbf{W}}^T\|_F^2 + \boldsymbol{\delta}(\mathbf{W}) + \frac{\rho}{2}\|\tilde{\mathbf{W}} - \mathbf{W}\|_2^2. \tag{3.25}$$

This guarantees a monotonic decrease of the objective function $\|\mathbf{A} - \tilde{\mathbf{U}}_t\tilde{\mathbf{W}}_t^T\|_F^2$. Each of these sub-problems is thus solved with ADMM separately, by alternating minimization steps of $\frac{1}{2}\|\mathbf{A} - \tilde{\mathbf{U}}\mathbf{W}_t^T\|_F^2 + \bar{\mathbf{U}}^T(\tilde{\mathbf{U}} - \mathbf{U}) + \frac{\rho}{2}\|\mathbf{U} - \tilde{\mathbf{U}}\|_2^2$ over $\tilde{\mathbf{U}}$ (***i***), with minimization steps of $\boldsymbol{\delta}(\mathbf{U}) + \frac{\rho}{2}\|\mathbf{U} - \tilde{\mathbf{U}}\|_2^2$ over $\mathbf{U}$ (***ii***), and gradient ascent steps (***iii***) on the dual variable $\bar{\mathbf{U}} \leftarrow \bar{\mathbf{U}} + (\tilde{\mathbf{U}} - \mathbf{U})$. A similar scheme is used for $\mathbf{W}$ updates. Step (***i***) is a simple convex quadratic program with equality constraints, whose KKT conditions (Karush, 1939; Kuhn and Tucker, 1951) yield a linear system with a Positive Semi-Definite (PSD) matrix. Step (***ii***) is a simple projection of $\tilde{\mathbf{U}}$ onto the convex set $\boldsymbol{\delta}^{-1}(\mathbf{0})$. Finally, step (***iii***) is inexpensive.

Concretely, we solved the quadratic program using Conjugate Gradient (Hestenes and Stiefel, 1952), from *jax.scipy.sparse.linalg.cg*. This indirect method only involves *matrix-vector* products and can be more GPU-efficient than methods that are based on matrix factorization (such as Cholesky decomposition). Also, we re-implemented the pseudo code of Huang et al. (2016b) in *Jax* for a fully GPU-compatible program. We used the primal variables $\mathbf{U}_0, \mathbf{W}_0$ returned by *sklearn.decompose.nmf* as a *warm start* for ADMM and observe that the high-quality initialization of these primal variables considerably speeds up the convergence of the dual variables.

3.3.2 Implicit differentiation of NMF

The Lagrangian of the NMF problem reads $\mathcal{L}(\mathbf{U}, \mathbf{W}, \bar{\mathbf{U}}, \bar{\mathbf{W}}) = \frac{1}{2}\|\mathbf{A} - \mathbf{U}\mathbf{W}^T\|_F^2 - \bar{\mathbf{U}}^T\mathbf{U} - \bar{\mathbf{W}}^T\mathbf{W}$, with dual variables $\bar{\mathbf{U}}$ and $\bar{\mathbf{W}}$ associated to the constraints $\mathbf{U} \geq 0, \mathbf{W} \geq 0$. It yields a function $\boldsymbol{F}$ based on the KKT conditions whose optimal tuple $\mathbf{U}, \mathbf{W}, \bar{\mathbf{U}}, \bar{\mathbf{W}}$ is a root.

For single NNLS problem (for example, with optimization over $\mathbf{U}$) the KKT conditions are:

$$\begin{cases} \nabla_{\mathbf{U}}\left(\frac{1}{2}\|\mathbf{A} - \tilde{\mathbf{U}}\tilde{\mathbf{W}}^T\|_F^2 + \bar{\mathbf{U}}^T(-\mathbf{U})\right) = 0, & \text{stationarity,} \\ -\mathbf{U} \leq 0, & \text{primal feasability,} \\ \bar{\mathbf{U}} \odot \mathbf{U} = 0, & \text{complementary slackness,} \\ \bar{\mathbf{U}} \geq 0, & \text{dual feasability.} \end{cases} \tag{3.26}$$

By stacking the KKT conditions of the NNLS problems the we obtain the so-called *optimality function* $\boldsymbol{F}$:

$$\boldsymbol{F}((\mathbf{U}, \mathbf{W}, \bar{\mathbf{U}}, \bar{\mathbf{W}}), \mathbf{A}) = \begin{cases} (\mathbf{U}\mathbf{W}^T - \mathbf{A})\mathbf{W} - \bar{\mathbf{U}}, & \text{stationarity} \\ (\mathbf{W}\mathbf{U}^T - \mathbf{A}^T)\mathbf{U} - \bar{\mathbf{W}}, & \text{stationarity} \\ \bar{\mathbf{U}} \odot \mathbf{U}, & \text{complementary slackness} \\ \bar{\mathbf{W}} \odot \mathbf{W}. & \text{complementary slackness} \end{cases} \tag{3.27}$$

The implicit function theorem allows us to use implicit differentiation to efficiently compute the Jacobians $\frac{\partial \mathbf{U}}{\partial \mathbf{A}}$ and $\frac{\partial \mathbf{W}}{\partial \mathbf{A}}$ without requiring to back-propagate through each of the iterations of the NMF solver:

$$\frac{\partial(\mathbf{U}, \mathbf{W}, \bar{\mathbf{U}}, \bar{\mathbf{W}})}{\partial \mathbf{A}} = -(\partial_1 \boldsymbol{F})^{-1} \partial_2 \boldsymbol{F}. \tag{3.28}$$

Implicit differentiation requires access to the dual variables of the optimization problem in equation 3.21, which are not computed by Scikit-learn's popular implementation. Scikit-learn uses Block coordinate descent algorithm (Cichocki and Phan, 2009; Févotte and Idier, 2011), with a randomized SVD initialization. Consequently, we leverage our implementation in Jax based on ADMM (Boyd et al., 2011).

Concretely, we perform a two-stage backpropagation *Jax (2)→Tensorflow (1)* to leverage the advantage of each framework. The lower stage (1) corresponds to feature extraction $\mathbf{A} = \boldsymbol{h}_l(\mathbf{X})$ from crops of images $\mathbf{X}$, and upper stage (2) computes NMF $\mathbf{A} \approx \mathbf{U}\mathbf{W}^T$. We use the *Jaxopt* library. The chain rule yields:

$$\frac{\partial \mathbf{U}}{\partial \mathbf{X}} = \frac{\partial \mathbf{A}}{\partial \mathbf{X}} \frac{\partial \mathbf{U}}{\partial \mathbf{A}}.$$

Remark 3.5. Backpropagation between autodiff frameworks

Usually, most Autodiff frameworks (e.g Tensorflow, Pytorch, Jax) handle the backpropagation step automatically. Unfortunately, combining two of those framework raised a new difficulty since they were not compatible at the time of the work (April 2022). In the meantime, Jax introduced a `callback` object to allow interoperability of the two frameworks. Below, we detail how to re-implement manually the two stages of auto-differentiation *without* relying on callbacks.

Since r is far smaller ($r = 25$ in all our experiments) than input dimension $\mathbf{X}$ (typically 224×244 for ImageNet images), back-propagation is the preferred algorithm in this setting over forward-propagation. We start by computing sequentially the gradients $\nabla_{\mathbf{X}} \mathbf{U}_i$ for all concepts $1 \leq i \leq r$. This amounts to compute $\boldsymbol{v} = \nabla_{\mathbf{A}} \mathbf{U}_i$ with Implicit Differentiation in Jax, convert the Jax array $\boldsymbol{v}$ into Tensorflow tensor, and then to compute $\nabla_{\mathbf{X}} \mathbf{U}_i = \frac{\partial \mathbf{A}}{\partial \mathbf{X}} \nabla_{\mathbf{A}} \mathbf{U}_i = \nabla_{\mathbf{X}}(\boldsymbol{h}_l(\mathbf{X}) \cdot \boldsymbol{v})$. The latter is easily done in Tensorflow. Finally we stack the gradients $\nabla_{\mathbf{X}} \mathbf{U}_i$ to obtain the Jacobian $\frac{\partial \mathbf{U}}{\partial \mathbf{X}}$.

3.3.3 Applications to Concept-Based XAI

NMF is famous for producing interpretable "basis of concepts" in various tasks (Lee and Seung, 1999). It can be applied in the latent space of a neural network. This produces a dictionary $\mathbf{W}$ of concepts, and coefficients $\mathbf{U}$. The concepts $\mathbf{W}$ can be understood as frequent features of the dataset that are recognized by the neural network. The coefficient $\mathbf{U}_i$ indicates how much the concept i is present in an image. This opens paths for the interpretability of the network decisions. This has been explored in Fel et al. (2023b). In this work, a concept $\mathbf{W}_i$ is characterized by the sets of *pixel patch* (of

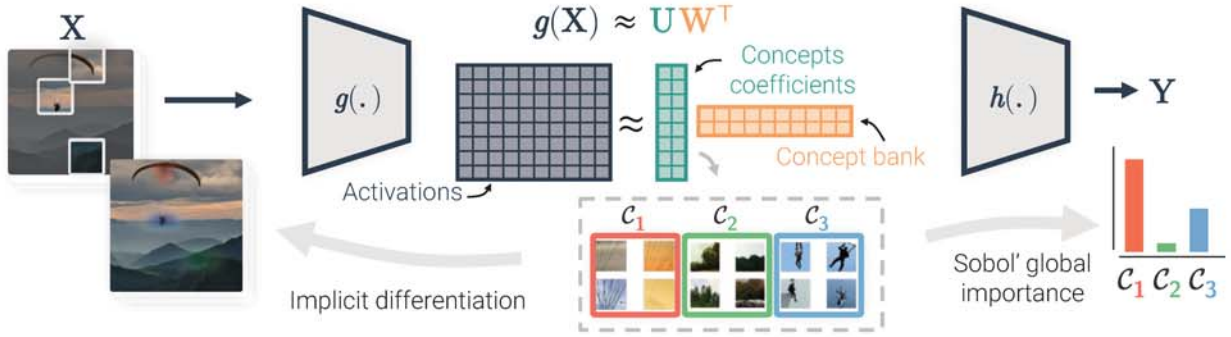


Figure 3.6: **Overview of CRAFT (Fel et al., 2023b)**, courtesy of Thomas Fel. Starting from a set of crops $\mathbf{X}$ containing a concept $\mathcal{C}$ (e.g., crops images of the class "parachute"), we compute activations $\boldsymbol{g}(\mathbf{X})$ corresponding to an intermediate layer from a neural network for random image crops. We then factorize these activations into two lower-rank matrices, ($\mathbf{U}$, $\mathbf{W}$). $\mathbf{W}$ is what we call a "concept bank" and is a new basis used to express the activations, while $\mathbf{U}$ corresponds to the corresponding coefficients in this new basis. We then extend the method with 3 new ingredients: (1) recursivity – by proposing to re-decompose a concept (e.g., take a new set of images containing $\mathcal{C}_1$) at an earlier layer, (2) a importance estimation using Sobol indices and (3) implicit differentiation to generate *concept attribution maps* to localize concepts in an image.

the train-set) that maximize the score $\mathbf{U}_i$. For a given image x, it is possible to extract the latent vector a with a feature extractor (e.g. the first layers of the network), to solve the Non Negative Least Square (NNLS) problem (with a fixed concept bank $\mathbf{W}$) to obtain the coefficients $\mathbf{U}_i$, and then to find *where* in the input image the concept has been detected, by back-propagating through NNLS problem. The algorithm is illustrated in Figure 3.6, reproduced here with the authorization of Thomas Fel. Concept-based XAI can be done with other dictionary learning methods, as explained in Fel et al. (2023a).

3.4 Conclusion

In this section, we shown that the paradigm of optimization as a layer could be used to enforce architectural constraints, such as:

- orthogonal weights in linear layers,
- sparse dictionaries with differentiable NMF layers.

When a certain structure needs to be enforced on the activations or the weights, it can be useful to think of these constraints as the solution to a well-chosen optimization problem.

In the three next chapters, we focus on *Lipschitz* constraints, with application to classification with generalization, robustness or privacy guarantees.

Chapter 4

Classification with Lipschitz constraints

> — Votre paramètre de température τ, là, on est bien d'accord que c'est la même chose que la constante de Lipschitz?
> — Oui, complètement. Mais présenté comme ça on était moins compris: les relecteurs déduisaient que petite constante de Lipschitz impliquait petite expressivité, et réciproquement. Ce qui est l'opposé de notre message, qui est plus subtil.
> — Vous vous êtes écrasés devant les reviewers, c'est dommage vous n'auriez pas dû.
>
> *Lesson of Wisdom taught by Rémi Flamary at NeurIPS Paris 2022.*

In this chapter, we study the expressiveness of Lipschitz functions in the context of supervised classification. This chapter is mostly adapted from the corresponding publication:

L. Béthune, T. Boissin, M. Serrurier, F. Mamalet, C. Friedrich, and A. G. Sanz. **Pay attention to your loss : understanding misconceptions about Lipschitz neural networks.**, Advances in Neural Information Processing Systems, *2022*.See Béthune et al. (2022).

Despite the competitiveness of Lipschitz-constrained networks over conventional networks on medium-scale problems (Cisse et al., 2017; Serrurier et al., 2021), Lipschitz constrained networks still suffer from misconceptions. A belief commonly invoked against networks of LipNet1 is that they are less expressive: "Lipschitz-based approaches suffer from some representational limitations that may prevent them from achieving higher levels of performance and applying to more complicated problems" (Huster et al., 2018). Some reviews of this early work contained similar criticism, showing that this belief is firmly rooted in the community. This objection is frequently encountered in .

Although this claim seems rational at first glance, the link between Lipschitz constant and expressiveness is not trivial. While there is an obvious lack of expressiveness for regression tasks, this intuition fades when it comes to classification. Indeed, every AllNet network $g : \mathbb{R}^n \to \mathbb{R}^K$ is L-Lipschitz for some (generally unknown) $L > 0$. Then $f = \frac{1}{L}g$ is a 1-Lipschitz neural network with the same decision boundary, since prediction $\arg\max_k g_k$ is invariant by positive rescaling of the logits. In particular, f has the same accuracy and also the same robustness to adversarial attacks as g. We illustrate this empirically by **training a LipNet1 network until it reaches train 99.96% accuracy on CIFAR-100 with random labels**.

Our goal is to demonstrate that, despite being empirically harder to train, LipNet1 networks are theoretically better grounded than AllNet networks when it comes to classification, through a threefold contribution on Expressiveness (Section 4.1), Robustness (Section 4.2) and Generalization (Section 4.3).

First, in Section 4.1 we confirm that LipNet1 are as expressive as AllNet networks for classification, and can learn arbitrary complex decision boundary. We show that hyper-parameters of the

loss are of crucial importance, and control the ability to fit properly the train set.

Then, in Section 4.2 we show that accuracy and robustness are often antipodal objectives. We characterize the robustness of the highest accuracy LipNet1 classifier: it is achieved by the Signed Distance Function. We also characterize the classifier of the highest certifiable robustness, and we show it corresponds to the dual potential of Wasserstein-1 distance.

Finally, in Section 4.3 we show that LipNet1 benefit from several generalization guarantees. They are consistent estimators: contrary to AllNet , we prove that their train loss will converge to test loss as the size of the train set increases. Moreover, we show that LipNet1 classifiers with margin are PAC-learnable (Valiant, 1984): it provides bounds on the number of train examples required to reach a targeted test accuracy. Interestingly, this bound is independent of the architecture size, which allows to training of enormous LipNet1 networks without risking overfitting.

We show in all three sections that tuning of losses hyper-parameters is crucial to control the tradeoff between train accuracy, certifiable robustness, and generalization gap.

This raises a new question: if constraining the Lipschitz constant does not prevent high accuracy, why is it difficult to train these LipNet1 networks? We answer the question through the prism of the loss function. We outline that the minimization of the popular cross-entropy (as it is done frequently in classification) is an ill posed problem for AllNet networks that may lead to catastrophic divergence of the weights, whereas it is well posed for networks of LipNet1 networks but fails to yield good test accuracy.

Therefore the main contributions of this chapter are to propose a general view of the multiple interest of LipNet1 in classification tasks, gathering known results, and demonstrating new ones.

Contents

4.1 Expressivity and the importance of the loss

First, we confirm that LipNet1 are as expressive as AllNet networks for classification, and can learn arbitrary complex decision boundary. We show that hyper-parameters of the loss are of crucial importance, and control the ability to fit properly the train set.

4.1.1 Boundary decision fitting

Proposition 2. Lipschitz Binary classification. *For any binary classifier $c : \mathcal{X} \to \mathcal{Y}$ with closed pre-images ($c^{-1}(\{y\})$ is a closed set) there exists a 1-Lipschitz function $f : \mathbb{R}^n \to \mathbb{R}$ such that $sign(f(x)) = c(x)$ on $\mathcal{X}$ and such that $\|\nabla_x f\| = 1$ almost everywhere (w.r.t Lebesgue measure).*

The proof of **Proposition 2** is constructive, we need to introduce the Signed Distance Function, already popularized in shape processing (Rousson and Paragios, 2002).

Definition 8. Signed Distance Function associated to decision boundary. *Let $c : \mathcal{X} \to \{-1, +1\}$ be any classifier with closed pre-images. Let $\bar{A} = \{x \in \mathbb{R}^n | c(x) = +1\}$ and $\bar{B} = \{x \in \mathbb{R}^n | c(x) = -1\} = \mathcal{X} \setminus \bar{A}$. Let $d(x, y) = \|x - y\|$ and $d(x, S) = \min_{y \in S} d(x, y)$ be the distance to a **closed** set S. Let $\partial = \{x \in \mathbb{R}^n | d(x, \bar{A}) = d(x, \bar{B})\}$. We define $f : \mathbb{R}^n \to \mathbb{R}$ as follow:*

$$f(x) = \begin{cases} d(x, \partial) & \text{if } d(x, \bar{B}) \geq d(x, \bar{A}) \\ -d(x, \partial) & \text{if } d(x, \bar{B}) < d(x, \bar{A}). \end{cases} \tag{4.1}$$

We denote by SDF(c) the function f.

The signed distance function f previously defined verifies all the properties, as a special case of Eikonal equation. We give the full proof here for completeness.

Proof. We start by proving that f is 1-Lipschitz. First, consider the case $d(x, \bar{B}) \geq d(x, \bar{A})$ and $d(y, \bar{B}) \geq d(y, \bar{A})$. Then we have $|f(x) - f(y)| = |d(x, \partial) - d(y, \partial)|$. Assume without loss of generality that $d(x, \partial) \geq d(y, \partial)$. Let $z \in \partial$ be such that $d(y, \partial) = d(y, z)$ (it is guaranteed to exist since ∂ is a closed set). Then by definition of $d(x, \partial)$ we have $d(x, z) \geq d(x, \partial)$. So:

$$|f(x) - f(y)| = |d(x, \partial) - d(y, \partial)| \leq d(x, z) - d(y, z) \leq d(x, y). \tag{4.2}$$

The cases $d(x, \bar{B}) < d(x, \bar{A})$ and $d(y, \bar{B}) < d(y, \bar{A})$ are identical. Now consider the case $d(x, \bar{B}) < d(x, \bar{A})$ and $d(y, \bar{B}) \geq d(y, \bar{A})$. Then we have $|f(x) - f(y)| = d(x, \partial) + d(y, \partial)$. We will proceed by contradiction. Assume that $d(x, \partial) + d(y, \partial) > d(x, y)$. Let $R > 0$ be such that $R < d(x, \partial)$ and $R + d(y, \partial) > d(x, y)$. We let:

$$z = x + \frac{R}{d(x, y)}(x - y).$$

Then we have $d(x, z) = \|\frac{R}{d(x,y)}(x - y)\| = \frac{R}{d(x,y)} d(x, y) = R < d(x, \partial)$. So by definition of ∂ we have $d(z, \bar{B}) < d(z, \bar{A})$. But we also have:

$$\begin{aligned} d(y, z) &= \|(x - y) + \frac{R}{d(x, y)}(x - y)\| = |1 - \frac{R}{d(x, y)}| \times \|x - y\| \\ &= |d(x, y) - R| < |d(y, \partial)| \text{ using the hypothesis on } R. \end{aligned} \tag{4.3}$$

So we have $d(z, \bar{B}) \geq d(z, \bar{A})$ which is a contradiction. Consequently, we must have $d(x, \partial) + d(y, \partial) \leq d(x, y)$. The function f is indeed 1-Lipschitz.

Now, we will prove that $\|\nabla_x f\| = 1$ everywhere it is defined. Let x be such that $y \in \arg\min_{y\in\partial} d(x,y)$ is unique. Consider $h = \epsilon\frac{(y-x)}{\|y-x\|}$ with $1 \geq \epsilon > 0$ a small positive real. We have $d(x, x+h) = \epsilon$, it follows by triangular inequality that $d(x+h, \partial) = d(x,\partial) - \epsilon$. We see that:

$$\lim_{\epsilon\to+\infty} \frac{f(x+h) - f(x)}{\|h\|} = -1.$$

The vector $u = -\nabla_x f$ is the (unique) vector for which $\langle u, \frac{f(x+h)-f(x)}{\|h\|}\rangle$ is minimal. Knowing that f is 1-Lipschitz yields that $\|\nabla_x f\| = 1$. For points x for which $\arg\min_{y\in\partial} d(x,y)$ is not unique, the gradient is not defined because different directions minimize $\langle u, \frac{f(x+h)-f(x)}{\|h\|}\rangle$ which contradicts the uniqueness of gradient vector. The number of points for which $y \in \arg\min_{y\in\partial} d(x,y)$ is not unique must have null measure, since Lipschitz functions are almost everywhere differentiable (by Rademacher's Theorem).

Finally, note that $\text{sign} f(x) = c(x)$ on $\bar{A}$ and $\bar{B}$. Indeed, in this case either $d(x,\bar{B}) < d(x,\bar{A})$ either $d(x,\bar{B}) > d(x,\bar{A})$ and the result is straightforward. □

With this proposition in mind, we can deduce Corollary 1.

Corollary 1 (LipNet1 is as powerful as AllNet for classification)**.** *For any neural network $f : \mathbb{R}^n \to \mathbb{R}$ there exists 1-Lipschitz neural network $\tilde{f} : \mathbb{R}^n \to \mathbb{R}$ such that sign$(f(x))$ = sign$(\tilde{f}(x))$.*

Proof. The proof sketched in Introduction is sufficient to show that LipNet1 networks and unconstrained ones have the same decision frontiers. We could have also taken a more convoluted path: take the classifier c associated to an AllNet network, consider the restriction to a subset $\mathcal{X}$ of the input space making the pre-images separated. Then we can apply Proposition 2 to get a 1-Lipschitz function with the same classification power, and finally approximate those functions (in the sense of uniform convergence) with LipNet1 network, thanks to the universal approximation theorem. □

For the multiclass case the label set is now $\mathcal{Y} = \{1, 2, \ldots, K\}$. In practice we use one-hot encoded vectors to compute the loss, by taking the $\arg\max_k$ over a vector of $\mathbb{R}^K$. J

Proposition 3 (Lipschitz Multiclass classification)**.** *For any multiclass classifier $c : \mathcal{X} \to \mathcal{Y}$ with closed pre-images there exists a 1-Lipschitz function $f : \mathbb{R}^n \to \mathbb{R}^K$ such that* $\arg\max_k f_k(x) = c(x)$ *on $\mathcal{X}$ and such that $\|J_x f\| = 1$ almost everywhere (w.r.t Lebesgue measure).*

The case $K > 2$ requires a slight change in the definition of signed distance function, to prove **Proposition 3**.

Definition 9 (Multiclass Signed Distance Function)**.** *Let $c : \mathcal{X} \to \{1, 2, \ldots K\}$ be any classifier with closed pre-images. Let $\bar{A_k} = c^{-1}(\{k\})$. Let $\partial = \{x \in \mathbb{R}^n | \exists k \neq l, d(x,\bar{A_k}) = d(x,\bar{A_l}) = \arg\min_m d(x,\bar{A_m})\}$. We define $f : \mathbb{R}^n \to \mathbb{R}^k$ as follow:*

$$f_k(x) = \begin{cases} d(x,\partial) & \text{if } d(x,\bar{A_k}) < d(x,\bar{A_l}) \text{ for all } l \neq k, \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

In overall the proof remains the same.

The level-sets of a $\text{Lip}_1(\mathcal{X}, \mathbb{R}^K)$ functions (and especially the decision boundary) can be arbitrarily complex: restraining classifiers to $\text{Lip}_1(\mathcal{X}, \mathbb{R})$ does not affect the classification power. The **Error**[1] of a classifier c is defined as $E(c) = \mathbb{E}_{(x,y)\sim\mathbb{P}_{XY}}[\mathbb{1}\{c(x) \neq y\}]$. The **Risk** of a classifier is defined as $\mathcal{R}(c) = E(c) - E(b)$ where b denotes the optimal Bayes classifier.

[1] Practitioners sometimes prefer to monitor accuracy $1 - E$.

Definition 10 (**ϵ-separated distributions**). *Distributions P and Q are ϵ-separated if the distance between supp P and supp Q exceeds $\epsilon > 0$.*

Corollary 2. Separable classes implies zero error. *If P and Q are ϵ-separated, then there exists a network $f \in LipNet1$ such that* ***error*** $E(sign \circ f) \coloneqq \mathbb{E}_{(x,y)\sim\mathbb{P}_{XY}}[\mathbb{1}\{sign(f(x)) \neq y\}] = 0$.

Proof. If classes are separable the optimal Bayes classifier b achieves zero error. Moreover, the topological closure $\overline{b^{-1}}(\{y\}), y \in \mathcal{Y}$ yields a set of closed sets that are all disjoints (since $\epsilon > 0$) and on which Proposition 3 can be applied, yielding a LipNet1 neural network with the wanted properties. □

Exemple 4.1. Bonus: non separable case.

We can also handle the case of non separable classes by imitating the optimal Baye classifier c. We take $\mathcal{X}$ a subset of the input space on which the pre-images of c are closed. The application of Proposition 2 for optimal Bayes classifier gives us a 1-Lipschitz function f with the same decision frontier as c. Finally, we can use the universal approximation theorem of Anil et al. (2019) to conclude there exists LipNet1 network that can approximate arbitrarily well the function f, and hence approximate arbitrarily well the classifier c on $\mathcal{X}$. Outside $\mathcal{X}$, the error is not controlled but depends on the volume of the set (supp $\mathbb{P}_X)/\mathcal{X}$ whose Lebesgue measure can be made arbitrarily small (by taking $\mathcal{X}$ big enough). As $\mathbb{P}_X$ admits a pdf w.r.t Lebesgue measure, then $\mathbb{P}_X((\text{supp } \mathbb{P}_X)/\mathcal{X})$ can be made arbitrarily small, and consequently the risk as well.

The class of LipNet1 networks does not suffer from bias for classification tasks. Some empirical studies show that indeed most datasets classes are separable (Yang et al., 2020) such as CIFAR10 or MNIST. Furthermore, even if the classes are not separable, functions of LipNet1 can nonetheless approximate the optimal Bayes classifier. Lipschitz constraint is **not** a constraint on the shape of the boundary (Figure 4.1), but rather on the slope of the landscape of f.

Exemple 4.2. Fractal decision boundary with Von Koch snowflake.

In figure 4.1 we plot the level set of the network f trained from the discretized ground truth (in 400×400 pixels) of the Signed distance function. The distance to the frontier ∂ is easily computed since the frontier ∂ is a finite collection of segments (fourth iteration of Von Koch snowflake fractal). We train a $128 \to 128 \to 128 \to 128 \to 128$ LipNet1 network. The network is trained with Mean Square Error (MSE), and we stop the training once the Mean Absolute Error (MAE) falls below 1.

4.1.2 Why Lipschitz networks are perceived as not expressive

LipNet1 networks cannot reach zero loss with BCE: this may explain why they are perceived as not expressive enough. Yet the minimizer of BCE exists and is well defined.

Proposition 4. BCE minimization for 1-Lipschitz functions. *Let $\mathcal{X} \subset \mathbb{R}^n$ be a compact and $\tau > 0$. Then the infimum in Equation 4.5 is a minimum, denoted $f^\tau \in Lip_1(\mathcal{X}, \mathbb{R})$:*

$$f^\tau \in \arg \inf_{f \in Lip_1(\mathcal{X},\mathbb{R})} \mathbb{E}_{(x,y)\sim\mathbb{P}_{XY}}[\mathcal{L}^{bce}_\tau(f(x), y)]. \tag{4.5}$$

Moreover, the LipNet1 networks will not suffer of vanishing gradient of the loss.

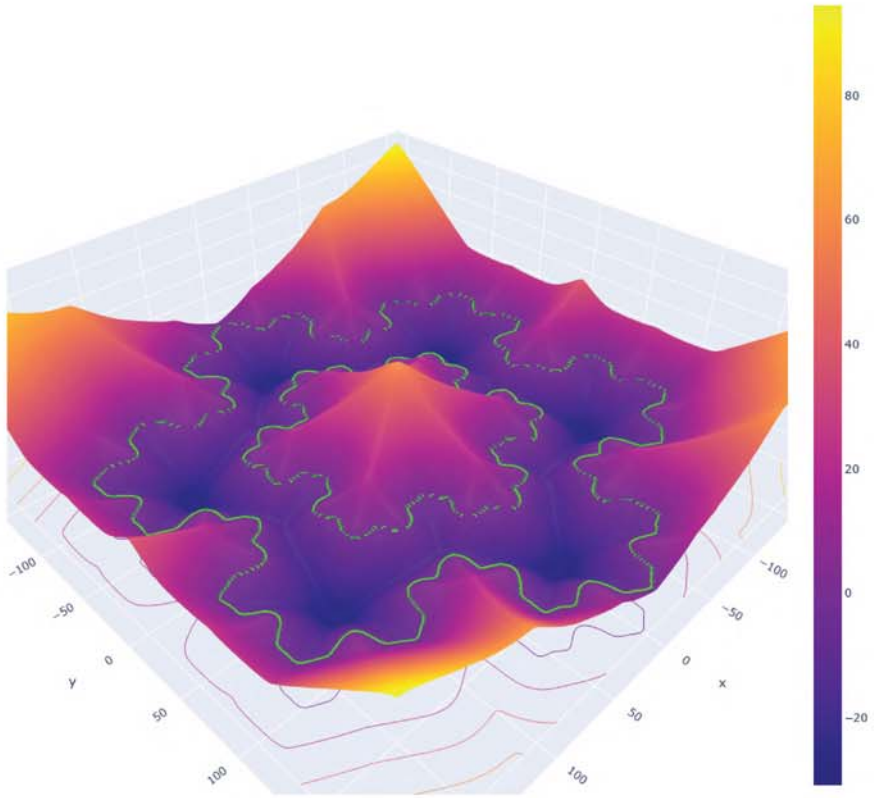


Figure 4.1: **Fractal Decision Boundary** ∂ with ∂ as the fourth iteration of Von Koch Snowflake. We chose P as the interior ring, while the center and the exterior correspond to Q. We train a LipNet1 network with Mean Square Error (MSE) to fit the Signed Distanc Function ground truth (160 000 pixels), until Mean Absolute Error (MAE) is inferior to 1. It proves empirically that LipNet1 networks can handle very sharp (almost fractal) decision boundary.

Proof. The proof is an application of Arzelà–Ascoli theorem.

Let $\mathcal{E}(f) = \mathbb{E}_{(x,y)\sim\mathbb{P}_{XY}}[\mathcal{L}(f(x),y)]$. Consider a sequence of functions f^t in $\mathrm{Lip}_L(\mathcal{X},\mathbb{R})$ such that $\lim_{t\to\infty} \mathcal{E}(f_t) = \inf_{f\in\mathrm{Lip}_L(\mathcal{X},\mathbb{R})} \mathcal{E}(f) = \mathcal{E}^*$.

Consider the sequence $u_t = \|f_t\|_\infty$. We want to prove that $(u_t)_{t\in\mathbb{N}}$ is bounded. Proceed by contradiction and observe that if $\limsup_{t\to\infty} u_t = +\infty$ then $\limsup_{t\to\infty} \mathcal{E}(f_t) = +\infty$. Indeed, for $\|f_t\|_\infty \geq 2L\mathrm{diam}\ \mathcal{X}$ we can guarantee that $\mathrm{sign} f_t$ is constant over $\mathcal{X}$ and in this case one of the two classes y is misclassified, knowing that $\lim_{f(x)\to\infty} \mathcal{L}(-yf(x),y) = \mathcal{O}(f(x)) \to +\infty$ yields the desired result. But if $\limsup_{t\to\infty} \mathcal{E}(f_t) = +\infty$, then $\mathcal{E}(f_t)$ cannot not converges to $\mathcal{E}^*$. Consequently, u_t must be upper bounded by some M.

Hence the sequence f_t is uniformly bounded. Moreover each function f_t is L-Lipschitz so the sequence f_t is uniformly equicontinuous. By applying Arzelà–Ascoli theorem we deduce that it exists a subsequence $f_{\phi(t)}$ (where $\phi : \mathbb{N} \to \mathbb{N}$ is strictly increasing) that converges uniformly to some f^*, and $f^* \in \mathrm{Lip}_L(\mathcal{X},\mathbb{R})$. As $\mathcal{E}(f^*) = \mathcal{E}^*$, the infimum is indeed a minimum. □

Warning 4.1. No element-wise vanishing gradients.

The upper bound on $\mathrm{Lip}(f)$ is turned into a lower bound on $\|\nabla_\theta \mathcal{L}(f_L^{\theta*}(x),y)\|$: there is no element-wise vanishing gradient. However its expectation $\|\nabla_\theta \mathbb{E}_{(x,y)\sim\mathbb{P}_{XY}}[\mathcal{L}((f_L^{\theta*}(x),y)]\| = 0$ is null at convergence. Therefore, the optimization of Lipschitz neural networks is truly a different beast than the ones of conventional networks: there is no "interpolation regime" like the ones required in Polyak step sizes (Loizou et al., 2021) or Armijo step sizes (Vaswani et al., 2019). The expectation is null, but the variance remains non-zero, *including at convergence*: this is problematic, as for small batch size, oscillations may be observed at convergence that impedes the final accuracy.

Proposition 5 (No vanishing BCE gradients)**.** *Let $(x_i,y_i)_{1\leq i\leq p}$ be a non trivial training set (i.e with more than one class) such that $x_i \in \mathcal{X}$, $\mathcal{X}$ a **bounded** subset of $\mathbb{R}^n$. Then there exists a constant $K > 0$ such that, for every minimizer f_L^* of BCE (known to exist thanks to Proposition 4) we have:*

$$f_L^* \in \arg\min_{f\in Lip_L(\mathcal{X},\mathbb{R})} \mathbb{E}_{(x,y)\sim\mathbb{P}_{XY}}[\mathcal{L}_T^{bce}(f(x),y)]. \tag{4.6}$$

And such that for every $1 \leq i \leq p$ we have the following:

$$|\frac{\partial}{\partial\tilde{y}}\mathcal{L}_T^{bce}(\tilde{y} = f_L^*(x_i),y_i)| \geq K. \tag{4.7}$$

Note that K only depends of the training set, not f_L^*.

Proof. Note that it exists $K' > 0$ such that $|f_L^*(x_i)| \leq K'$ for all x_i and all minimizers f_L^*, just like in the proof of Proposition 4, because otherwise we could exhibit a sequence of minimizers $(f_L^*)_t$ not uniformly bounded, which is a contradiction. Consequently $|\frac{\partial}{\partial\tilde{y}}\mathcal{L}_T^{bce}(\tilde{y} = f(x_i),y_i)| \geq \frac{1}{1+\exp(|f(x_i)|)} \geq \frac{1}{1+\exp(K')} = K$. □

It means that a non-null gradient will remain for each element-wise gradient, but their mean over the train set after convergence will be the null vector. Consequently, we must expect high variance in gradients and oscillations when we get closer to the minimum.

Machine learning practitioners are mostly interested in maximizing accuracy. However, the minimizer of BCE is not necessarily a minimizer of the error (see Figure 4.2). Yet, BCE is notoriously

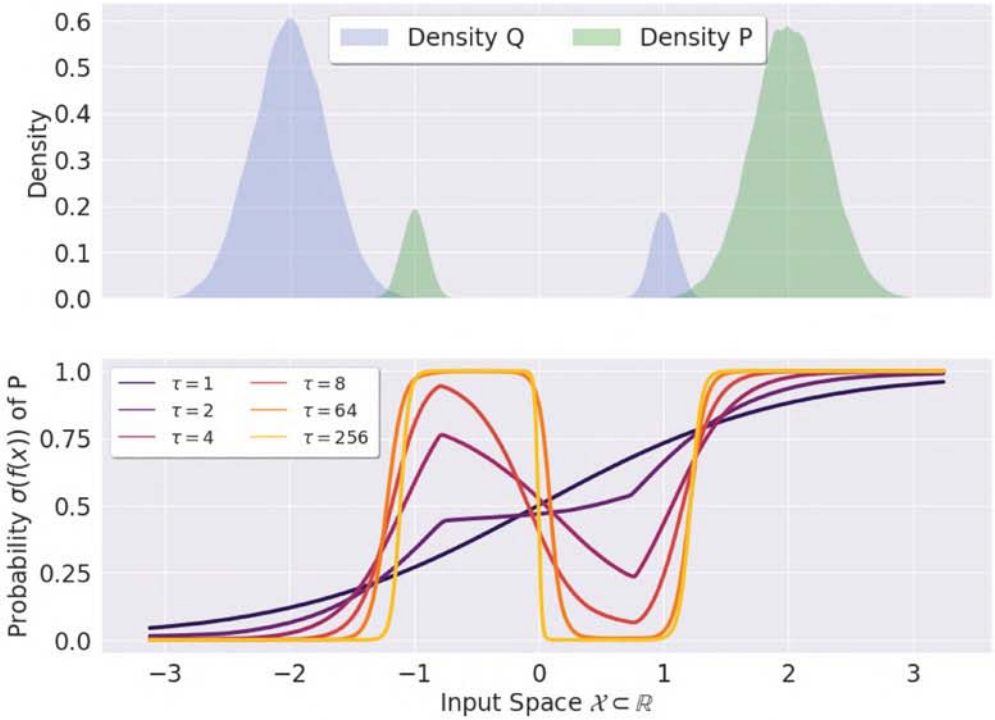


Figure 4.2: **Importance of τ in BCE**. We train a LipNet1 network with BCE and different values for τ. We chose a toy example where P and Q are Gaussian mixtures with two modes of weights 0.9 and 0.1. We highlight the different shapes of the minimizer $\sigma \circ f$ as function of τ. **High values of τ leads to better fitting, whereas for lower τ the small weights Gaussian of the mixture are treated as noise and ignored**.

a differentiable proxy of the error $E(\text{sign} \circ f)$, and as $\tau \to \infty$ we get asymptotically closer to maximum empirical accuracy. Bigger value for τ might ultimately lead to overfitting, playing the same role as the Lipschitz constant L (see Figure 4.2).

The implicit parameter $\tau = 1$ of the loss is partially responsible of the poor accuracy of LipNet1 networks in literature, and not by any means the hypothesis space LipNet1 itself. This can be observed in practice : when temperature τ (resp. margin m) of cross-entropy (resp. hinge loss) is correctly adjusted a small LipNet1 CNN can reach a competitive **88.2% validation accuracy on the CIFAR-10 dataset** (results synthetized and discussed in Figure 4.5) *without* residual connections, batch normalization or dropout. Conversely, AllNet networks are roughly equivalent to learning a LipNet1 network with $\tau \to \infty$: without regularization or data augmentation, such a network can always reach 100% train accuracy without generalization guarantees.

Exemple 4.3. Fitting CIFAR100 with random labels.

This experiment illustrates that constraining the Lipschitz of a network does not affect its expressive power. To show this we train a constrained network on the CIFAR100 dataset where all labels have been replaced with random labels. This task is now a widely recognized benchmark to evaluate the expressiveness of an architecture (Zhang et al., 2021b). The architecture of this network is as simple as possible: two orthogonal dense layers with 1024 neurons are followed by a dense layer (not orthogonal but with unit norms rows), biases in

every linear transformation, and GroupSort2 activation.

Loss	$\mathcal{L}^{bce}_{\tau=256}$	$\mathcal{L}^{hkr}_{\alpha=256,m=36/255}$
Clean Accuracy	99.9%	99.8%
Certifiable accuracy at $\epsilon = 36$	38.2%	91.0%
Certifiable accuracy at $\epsilon = 72$	21%	19%

At first glance it mights seem surprising to see both high accuracy and high provable robustness on a dataset with random labels. This is compliant with the idea expressed by the authors of Yang et al. (2020): for a given accuracy one can increase the robustness radius around a sample x_1 up to the value $\|\frac{x_1-x_2}{2}\|_2$ where x_2 is the closest sample with a different label. The decision frontier is close to the decision frontier of the 1-nearest neighbor based on the trained set. This illustrates that constraining the Lipschitz constant does not necessarily decrease accuracy and does not necessarily increase robustness.

Takeaways

Our empirical observation can be summarized as follow.
We let:

$$f^{\infty} \in \arg \inf_{f \in \text{AllNet}} \mathbb{E}_{(x,y)\sim\mathbb{P}_{XY}}[\mathcal{L}^{bce}_{\tau}(f(x), y)]. \tag{4.8}$$

Then the following inequality holds in general:

$$f^{\tau=1} \neq \frac{1}{\text{Lip}(f^{\infty})} f^{\infty}. \tag{4.9}$$

This can be informally reformulated as "Optimizing over the set of 1-Lipschitz functions, or re-normalizing a function to make it 1-Lipschitz does not yield the same decision frontier". In fact, f^{∞} might not be well defined as we will see in section 4.3.1.

4.2 Robustness guarantees and link to optimal transport

Here, we show that accuracy and robustness are often antipodal objectives. We characterize the robustness of the highest accuracy LipNet1 classifier: it is achieved by the signed distance function. We also characterize the classifier of highest certifiable robustness, and we show it corresponds to the dual potential of Wasserstein-1 distance (i.e the discriminator of a WGAN (Arjovsky et al., 2017)).

Is there a trade-off between accuracy and robustness? Although the existence of a trade-off between accuracy and robustness is commonly admitted, some works argue that "Robustness is not inherently at odds with accuracy"(Yang et al., 2020). We propose a unified consideration by stating that for a given train accuracy, robustness can be maximized up to a certain point, but allowing a lower train accuracy helps achieving a higher robustness. Finally one must keep in mind that this trade-off lives in the shade of generalization (see Section 4.3).

4.2.1 Improving the robustness of the maximally accurate classifier

The Signed Distance Function (Rousson and Paragios, 2002) (SDF) associated to the frontier ∂ of Bayes classifier b is the 1-Lipschitz function that provides the largest certificates among the classifiers

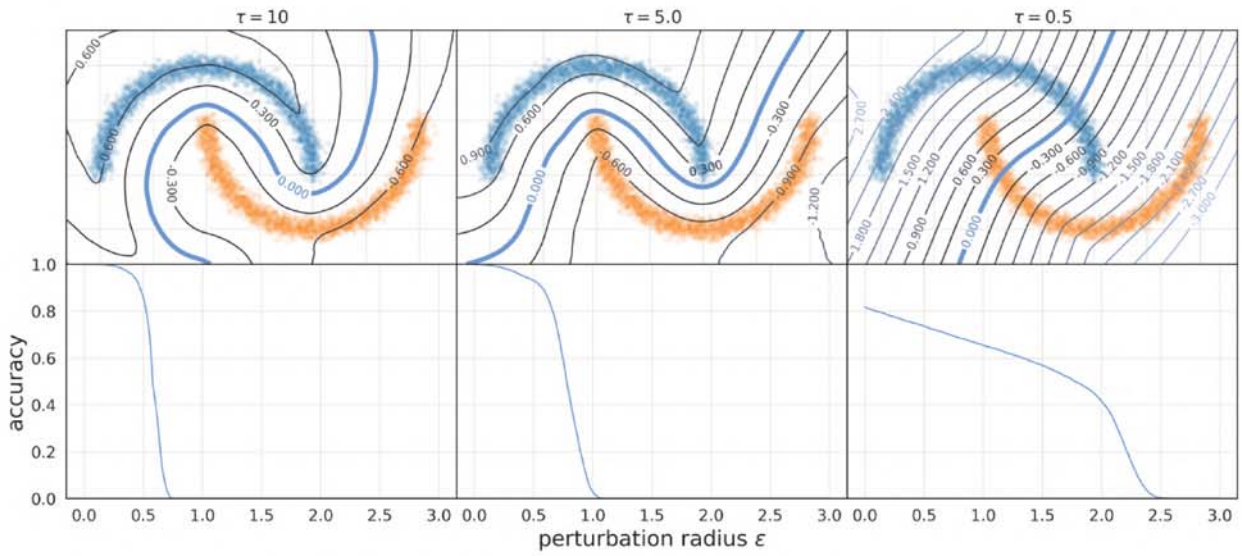


Figure 4.3: **Accuracy-robustness tradeoff:** Each network is optimal with respect to a certain criterion. The leftmost network is the most accurate at robustness radius $\epsilon \leq 0.3$, the rightmost maximizes the MCR at the cost of low clean accuracy. The center network corresponds to a compromise.

of maximum accuracy.

Corollary 3. *For the SDF(b), the bound of Property 1 is tight:* $\epsilon = |f(x)|$. *In particular* $\delta = -f(x)\nabla_x f(x)$ *is guaranteed to be an adversarial attack. The risk is the smallest possible. There is no classifier with the same risk and better certificates. Said otherwise the SDF(b) is the solution to:*

$$\max_{f \in Lip_1(\mathbb{R}^n, \mathbb{R})} \min_{x \in \mathcal{X}} \min_{\substack{\delta \in \mathbb{R}^n \\ sign(f(x+\delta)) \neq sign(f(x))}} \|\delta\|, \quad \text{under the constraint } f \in \operatorname*{arg\,min}_{g \in Lip_1(\mathbb{R}^n, \mathbb{R})} E(sign \circ g). \tag{4.10}$$

where $\mathcal{R}(c) = E(c) - E(b)$ *is the risk of the classifier, and where* b *denotes the optimal Bayes classifier.*

Proof. Those properties hold by construction. The risk $\mathcal{R}(\text{sign}(f))$ is minimal since f is build with the optimal Bayes classifier. Note that, in general, for any classifier $c : \mathcal{X} \to \mathcal{Y}$ the bound of Property 1 is tight by construction for SDF(c). Indeed $f(x)$ is the distance to the frontier, and the direction is given by $\nabla_x f(x)$. □

Those certificates are exactly equal to the distance of adversarial samples. Iterative gradient based attacks (see Chakraborty et al. (2021) and references therein) can succeed in one step: therefore empirical attacks tend to provide the same (optimal) adversarial examples, which was also noticed previously in Serrurier et al. (2021). Far from being a weakness, this may improve the interpretability of the model (Dong et al., 2017; Tomsett et al., 2018; Ross and Doshi-Velez, 2018).

The SDF(b) cannot be explicitly constructed since it relies on the (unknown) optimal Bayes classifier. We deduce that to train a LipNet1 network that yields the best robustness certificates, we must aim to maximize $|f(x)|$ over the train set (with appropriate sign depending on the label).

4.2.2 Improving the accuracy of the maximally robust classifier

On the opposite side, we exhibit a family of classifiers with lower accuracy but with higher certifiable robustness. We insist that the quantity of interest is the *certifiable robustness* $|f(x)|$ and not the *true empirical robustness* ϵ (which can be higher). The former is computed exactly and freely, while the latter is a difficult problem for which only upper bounds returned by attacks are available. In the literature, the robustness is only evaluated on well classified examples. The certificate can be both interpreted as a form of "confidence" of the network, and as the minimal perturbations required to switch the class. Hence, we shall weight negatively this certificate for the examples that are misclassified since confidence in presence of errors is worse. For this reason, we propose in Definition 11 a new metric called the Mean Certifiable Robustness (MCR).

Definition 11 (**Mean Certifiable Robustness – MCR**). *For any function $f : \mathcal{X} \to \mathbb{R} \in$ LipNet1 we define its weighted mean certifiable robustness $\mathcal{R}_{(P,y)}(f)$ on class P with label $y \in \{-1, +1\}$ as:*

$$\begin{aligned}\mathcal{R}_{(P,y)}(f) :=& \mathbb{E}_{x\sim P}[\mathbb{1}\{yf(x) > 0\}|f(x)|] + \mathbb{E}_{x\sim P}[-\mathbb{1}\{yf(x) < 0\}|f(x)|] \\ =& \mathbb{E}_{x\sim P} yf(x).\end{aligned} \tag{4.11}$$

We can readily see from the definition that the classifier with highest MCR for class P is the constant classifier $f = y \times \infty$. The interest of this notion arises when we consider minimizing the loss function $\mathcal{L}^W(f(x), y) := -yf(x)$ for different classes P and Q, i.e when looking for classifier with the highest MCR.

Property 4. Wasserstein classifiers (i.e WGAN discriminators) are optimally robust. *The minimum of $\mathcal{L}^W(f(x), y)$ over P and Q is the Wasserstein-1 distance (Villani, 2008) between P and Q according to the Kantorovich-Rubinstein duality:*

$$\max_{f\in Lip_1(\mathcal{X},\mathbb{R})} \mathcal{R}_{(P,+1)}(f) + \mathcal{R}_{(Q,-1)}(f) = \min_{f\in Lip_1(\mathbb{R}^n,\mathbb{R})} \mathbb{E}_{\mathbb{P}_{XY}}[\mathcal{L}^W(f(x), y)] = \mathcal{W}_1(P, Q). \tag{4.12}$$

Proof. The result is straightforward by writing the dual formulation (following Kantorovich-Rubinstein) of Wasserstein $\mathcal{W}_1$ metric. By Remark 6.3 of Villani (2008) the Wasserstein-1 distance is the Kantorovich-Rubinstein distance:

$$\mathcal{W}_1(P, Q) = \sup_{f\in \mathrm{Lip}_1(\mathcal{X},\mathbb{R})} \mathbb{E}_{x\sim P}[f(x)] + \mathbb{E}_{z\sim Q}[f(z)].$$

We see that:

$$\begin{aligned}\mathcal{W}_1(P, Q) &= \sup_{f\in \mathrm{Lip}_1(\mathcal{X},\mathbb{R})} \mathbb{E}_{x\sim P}[f(x)] - \mathbb{E}_{z\sim Q}[f(z)] \\ &= \inf_{f\in \mathrm{Lip}_1(\mathcal{X},\mathbb{R})} \mathbb{E}_{x\sim P}[-f(x)] + \mathbb{E}_{z\sim Q}[-(-f(z))] \\ &= \inf_{f\in \mathrm{Lip}_1(\mathcal{X},\mathbb{R})} \mathbb{E}_{(x,y)\sim \mathbb{P}_{XY}}[\mathcal{L}^W(f(x), y)].\end{aligned} \tag{4.13}$$

By Kirszbraun's theorem, the optimum of Equation 4.13 can be extended into a 1-Lipschitz function over $\mathbb{R}^n$. This function can, in turn, be approximated by a LipNet1 network over the domain of interest. □

Even though the minimizer of $\mathcal{L}_W(f(x), y)$ can have low accuracy, it has the highest MCR. Interestingly, the minimizer f^* of equation 4.12 is invariant by translation: $f^* - T$ is also a minimizer for any $T \in \mathbb{R}$. When $T \to \infty$ (resp. $-\infty$) the classifier has 100% recall on Q (resp. P), and 0%

on P (resp. Q). Does it always exist T^* with 100% accuracy overall? Sadly, even when the P and Q have disjoint support, the answer is no. We precise this empirical observation of Serrurier et al. (2021) in Proposition 6.

Proposition 6. WGAN discriminators are weak classifiers. *For every $\frac{1}{2} \geq \epsilon > 0$ there exist distributions P and Q with disjoint supports in $\mathbb{R}$ such that for any optimum f of equation 4.12, the error of classifier $\text{sign} \circ f$ is superior to $\frac{1}{2} - \epsilon$.*

Proof. We will build P and Q as a finite collection of Diracs. Let $P = \frac{1}{n}\sum_{i=1}^{n} \delta_{4(i-1)}$ and $Q = \frac{1}{n}\sum_{i=1}^{n} \delta_{4i-1}$ for some $n \in \mathbb{N}$, where δ_x denotes the Dirac distribution in $x \in \mathbb{R}$. A example is depicted in Figure 4.4 for $n = 20$. In dimension one, the optimal transportation plan is easy to compute: each atom of mass from P at position i is matched with the corresponding one in Q to its immediate right.

Consequently we must have $f(4i-1) = f(4(i-1))+3$. The function f is not uniquely defined on segments $[4i-1, 4i]$ but it does not matter: since f is 1-Lipschitz we must have $|f(4i-1) - f(4i)| \leq 1$. Consequently in every case for $i < j$ we must have $f(4(i-1)) < f(4(j-1))$ and $f(4i-1) < f(4j-1)$. Said otherwise, f is strictly increasing on supp P and supp Q. The solutions of the problems are invariant by translations: if f is the solution, then $f - T$ with $T \in \mathbb{R}$ is also a solution. Let's take a look at classifier $c(x) = \text{sign}(f(x) - T)$. If T is chosen such that $f(4(i-1))-T < 0$ and $f(4i-1)-T > 0$ for some $1 \leq i \leq n$ then $(n-1)+2 = n+1$ points are correctly classified on a total of $2n$ points. It corresponds to an error of $\frac{n-1}{2n} = \frac{1}{2} - \frac{1}{2n}$. We see that other values for T leads to worse error. Take $n = \lceil \frac{1}{2\epsilon} \rceil$ to conclude. □

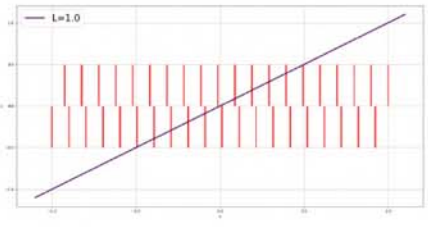


Figure 4.4: Pathological distributions P and Q of 20 points each, on which the accuracy of the Wasserstein minimizer cannot exceed 52.5%.

Note that the minimum of Equation 4.12 is also invariant by dilatation: any *finite* upper bound L can be chosen.

4.2.3 Controlling of the accuracy/robustness tradeoff

Now that the extrema of the accuracy robustness tradeoff were characterized in 4.2.1 and 4.2.2, is yet to be answered if it is possible to control this tradeoff using conventional loss (and its parameters, as introduced in 4.1.2).

Interestingly, observe that $\mathcal{L}_\tau^{bce}(f(x), y) = \log 2 - \frac{y\tau f(x)}{2} + \mathcal{O}(\tau^2 f^2(x))$ so when $\tau \to 0$ we get:

$$\min_{f \in \text{Lip}_1(\mathcal{X}, \mathbb{R})} \frac{4}{\tau}\left(\mathbb{E}_{(x,y)\sim P_{XY}}[\mathcal{L}_\tau^{bce}(f(x), y)] - \log 2\right) = -\mathcal{W}_1(P, Q).$$

In the limit of small temperatures, the BCE minimizer essentially behaves like the classifier of the highest MCR (see Figure 4.5). Similarly, the HKR loss $\mathcal{L}^{hkr}$ introduced in Serrurier et al. (2021) for LipNet1 training allows fine grained control of the accuracy-robustness tradeoff:

$$\mathcal{L}_{m,\alpha}^{hkr}(f(x), y) = \mathcal{L}^W(f(x), y) + \alpha\mathcal{L}_m^H(f(x), y) = -yf(x) + \alpha \max(0, m - yf(x)). \tag{4.14}$$

We recover $\mathcal{W}_1$ behavior for $\alpha = 0$, and hinge $\mathcal{L}_m^H$ behavior for $\alpha \to \infty$, in a fashion that reminds the role of τ for $\mathcal{L}^{bce}$.

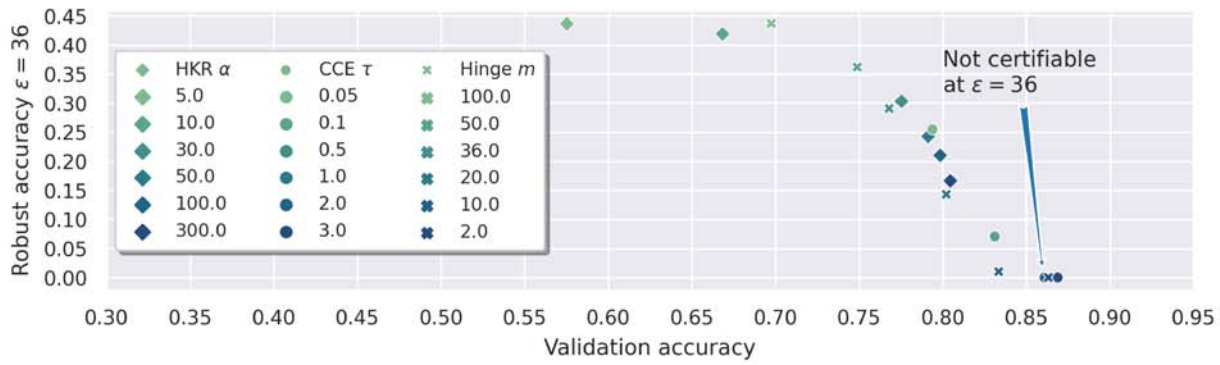


Figure 4.5: **Accuracy-Robustness trade-off on CIFAR10 with Hinge, HKR and Categorical Cross-Entropy (CCE) hyper-parameters.** Overall, for a given network architecture, a Pareto front appears between clean accuracy and robust accuracy. We move along it by tuning the parameters of each loss. We trained small LipNet1 *CNNs* (0.4M params) with basic data augmentation.

Takeaways

The key takeaway is that BCE, HKR and hinge loss have parameters that allow to control the accuracy/robustness tradeoff, reaching on one side the maximum robustness of Mean Certifiable Robustness, and the accuracy of unconstrained networks on the other. Empirically this tradeoff is observed as a Pareto front with accuracy on one axis, and robustness on the other. Figure 4.5 shows this on the CIFAR10 dataset using the accuracy at $\epsilon = 36/255$ as a measure of robustness.

A more fine-grained analysis of Binary Cross-Entropy

In the following, we try to draw other links between BCE minimization and optimal transport. Since the objective function is optimized with gradient descent, the gradients of the loss is the object of interest. We re-introduce f_θ as a function parameterized by θ, mapping the input to the logits. Let $g^p_\theta(x) = \sigma(f_\theta(x))$ and $g^q_\theta(x) = 1 - \sigma(f_\theta(x))$. $g^p_\theta(x)$ (resp. $g^q_\theta(x)$) are the predicted probabilities of class +1 (resp. -1).

Now define $\mathcal{Z}^p_\theta = \mathbb{E}_{x\sim P}[g^q_\theta(x)]$ and $\mathcal{Z}^q_\theta = \mathbb{E}_{x\sim Q}[g^p_\theta(x)]$. $\mathcal{Z}^p_\theta$ can be seen as the weighted rate of **false negatives**. That is the average mass of probability given to class -1 by f_θ when examples are sampled from class $+1$. Similarly, $\mathcal{Z}^q_\theta$ can be seen as the rate of **false positives**. We let:

$$\mathrm{d}P_\theta(x) = \frac{1}{\mathcal{Z}^p_\theta} g^q_\theta(x)\,\mathrm{d}P(x) \text{ and } \mathrm{d}Q_\theta(x) = \frac{1}{\mathcal{Z}^q_\theta} g^p_\theta(x)\,\mathrm{d}Q(x). \tag{4.15}$$

Consequently, P_θ (resp. Q_θ) is a valid probability distribution on $\mathbb{R}^n$ corresponding to the probability of an example x to be incorrectly classified in class -1 (resp. $+1$). With these notations, the full expression of the gradient takes a simple form. Behold the minus sign: it is a gradient *descent* and not a gradient *ascent*.

$$-\nabla_\theta \left(\mathbb{E}_{x\sim P}[\mathcal{L}(f_\theta(x), +1)] + \mathbb{E}_{x\sim Q}[\mathcal{L}(f_\theta(x), -1)]\right) = \mathcal{Z}^p_\theta \mathbb{E}_{x\sim P_\theta}[\nabla_\theta f_\theta(x)] - \mathcal{Z}^q_\theta \mathbb{E}_{x\sim Q_\theta}[\nabla_\theta f_\theta(x)] \tag{4.16}$$

We apply a bias term $T \in \mathbb{R}$ to classify with $f_\theta - T$ instead. For a well-chosen T we can enforce $\mathcal{Z}^p_\theta = \mathcal{Z}^q_\theta$, and such T can be found using the bisection method. The optimization is performed over

the set of 1-Lipschitz functions. We end up with:

$$\mathcal{Z}_\theta^p(\mathbb{E}_{x\sim P_\theta}[\nabla_\theta f_\theta(x)] - \mathbb{E}_{x\sim Q_\theta}[\nabla_\theta f_\theta(x)]). \tag{4.17}$$

This is the gradient for the computation of Wasserstein metric $\mathcal{W}$ between P_θ and Q_θ, using Rubinstein-Kantorovich dual formulation. Hence, binary cross-entropy minimization is similar to the computation of a transportation plan between "errors" distributions P_θ and Q_θ. Note that P_θ and Q_θ depend on the current classifier $f_\theta - T$, so the problem is not stationary as the training progresses.

In AllNet networks, as the training proceeds, the Lipschitz constant increases (equivalently increasing τ) and the loss "self-correct" with P_θ and Q_θ to improve accuracy.

Takeaways

This reasoning can be generalized to any elementwise loss in symmetric binary classification tasks. In this context, symmetric means that the sign of the loss is flipped under label swapping. For suitable weight functions g_θ^p and g_θ^q, the "importance sampling" trick used previously can be re-applied, and always yield gradient steps of the form:

$$\mathcal{Z}_\theta^p(\mathbb{E}_{x\sim P_\theta}[\nabla_\theta f_\theta(x)] - \mathbb{E}_{x\sim Q_\theta}[\nabla_\theta f_\theta(x)]). \tag{4.18}$$

for some some distributions P_θ and Q_θ. Here, P_θ and Q_θ can be seen as a dynamic re-weighting of the samples on the fly, in a scheme that reminds of *boosting* approaches (see for example Friedman (2001) and Friedman (2002)). This also draws some links with the re-weighted gradient descent algorithms like those studied in El Hanchi et al. (2022) or Kumar et al. (2023). In either case, if the loss depends on a hyper-parameter τ such that $P_\theta, Q_\theta \xrightarrow[\tau\to 0]{} c$ for some constant c, then in the limit the gradient step behaves exactly like the one of Kantorovich-Rubinstein loss.

4.3 Generalization results

These last two sections demonstrated that restraining networks to be in LipNet1 does not impact the classification capabilities while providing certificates of robustness; however, for these networks the loss parameters play an important role in this trade-off.

In this section, we explore the statistical and optimization properties of LipNet1 networks, and we prove the assumption of Gouk et al. (2021) that "adjusting the Lipschitz constant of a feed-forward neural network controls how well the model will generalise to new data".

4.3.1 Consistency of LipNet1 class

LipNet1 class enjoys another remarkable property since it is a Glivenko-Cantelli class: minimizers of Lipschitz losses are consistent estimators. In other words, as the size of the training set increases, the training loss becomes a proxy for the test loss: LipNet1 neural networks will not overfit in the limit of (very) large sample sizes.

Proposition 7. Train Loss is a proxy of Test Loss. *Let $\mathbb{P}_{XY}$ a probability measure on $\mathcal{X}\times\mathcal{Y}$ where $\mathcal{X}\subset\mathbb{R}^n$ is a bounded set. Let $(x_i, y_i)_{1\leq i\leq p}$ be a sample of p iid random variables with law*

$\mathbb{P}_{XY}$. *Let $\mathcal{L}$ be a Lipschitz loss function over $\mathbb{R} \times \mathcal{Y}$. We define:*

$$\mathcal{E}_p(f) := \frac{1}{p}\sum_{i=1}^{p} \mathcal{L}(f(x_i), y_i) \text{ and } \mathcal{E}_\infty(f) := \mathbb{E}_{(x,y)\sim\mathbb{P}_{XY}}[\mathcal{L}(f(x), y)]. \tag{4.19}$$

Then the empirical loss $\mathcal{E}_p(f)$ converges to the test loss $\mathcal{E}_\infty(f)$ (taking the limit $p \to \infty$):

$$\min_{f\in Lip_L(\mathcal{X},\mathbb{R})} \mathcal{E}_p(f) \xrightarrow{a.s} \min_{f\in Lip_L(\mathcal{X},\mathbb{R})} \mathcal{E}_\infty(f). \tag{4.20}$$

Proof. This result is an application of Glivenko-Cantelli theorem. We proved in Proposition 4 that the minimum of equation 4.5 is attained, so we replace inf by min for the Lipschitz loss function $\mathcal{L}$. We restrict ourselves to a subset of $\mathrm{Lip}_L(\mathcal{X}, \mathbb{R})$ on which $\|f\|_\infty \leq 2L\mathrm{diam}\ \mathcal{X}$ because the minimum lies in this subspace. We have:

$$|\min_f \mathcal{E}_p(f) - \min_f \mathcal{E}_\infty(f)| \leq \max_f |\mathcal{E}_p(f) - \mathcal{E}_\infty(f)|.$$

Let $g_y(x) = \mathcal{L}(f(x), y)$. Note that g is also Lipschitz and bounded on $\mathcal{X}$. The *entropy with bracket* (see A.W. van der vaart (1996), Chapter 2.1) of the class of functions $\mathcal{G} = \{g_y = \mathcal{L} \circ f | f \in \mathrm{Lip}_L(\mathcal{X}, \mathbb{R}), y \in \mathcal{Y}, \mathcal{X}$ bounded and $\|f\|_\infty \leq 2L\mathrm{diam}\ \mathcal{X}\}$ is finite (see A.W. van der vaart (1996), Chapter 3.2). Consequently $\mathcal{G}$ is Glivenko-Cantelli. Finally $\max_f |\mathcal{E}_p(f) - \mathcal{E}_\infty(f)| \xrightarrow{a.s} 0$ which concludes the proof.

Results of **Table 4.2**. Loss $\mathcal{L}^{hkr}_{m,\lambda}$ still belong to Glivenko-Cantelli classes as sum of functions $\mathcal{L}^W$ and $\mathcal{L}^H_m$ from Glivenko-Cantelli classes (on same distribution $\mathbb{P}_X$). □

It is another flavor of the bias-variance trade-off in learning. Thanks to Corollary 2 we know the LipNet1 class does not suffer of bias, while the generalization gap (i.e the variance) can be made as small as we want by increasing the size of the training set (see Figure 4.6). This result may seem obvious, but we emphasize **this property is not shared by AllNet networks**. Nonetheless, most practitioners take for granted that bigger training sets ensure generalization for AllNet networks.

Exemple 4.4. experimental protocol of figure 4.6.

As the size of the training set increases, the training loss becomes a proxy for the test loss. However, we do not give convergence speed bounds: we do not know how many samples are needed for a given task to observe the convergence between train and test losses. Moreover, the losses are parametrized (e.g by τ, α, m) so we expect to have different convergence rates, depending on those parameters. In order to observe this empirically on the CIFAR10 dataset, the same architecture was trained successively on 2%, 5%, 10%, 25%, 50% and 100% of the dataset. The sub-sampling was performed with a different seed each time. Similarly, this procedure has been repeated with different values for τ. The number of examples required to close the generalization gap is dataset specific in general, however it seems that with low τ fewer examples are required.

4.3.2 Divergence and overfitting in conventional networks

Surprisingly, on AllNet networks, minimization of BCE leads to uncontrolled growth of Lipschitz constant and saturation of the predicted probabilities. This is an impediment to generalization results.

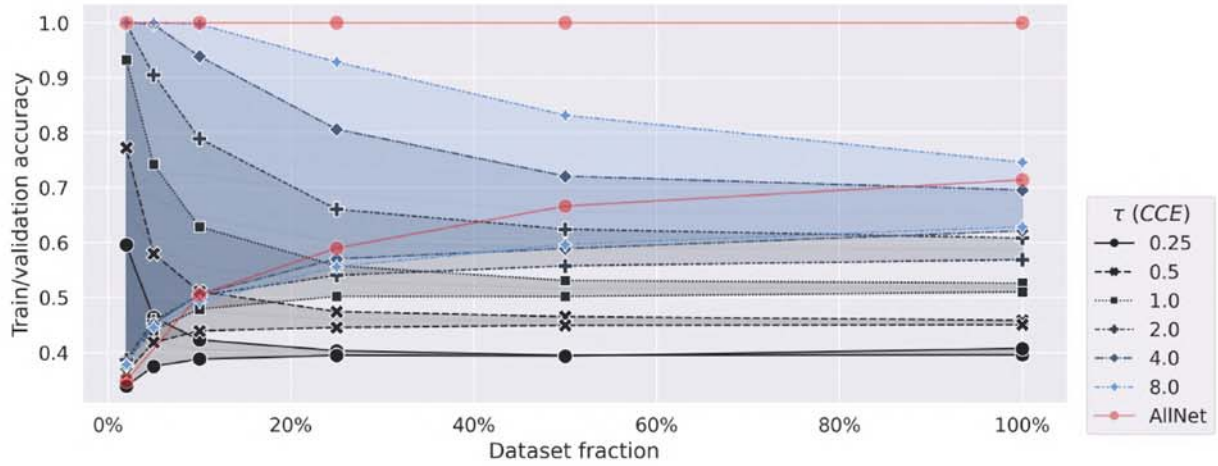


Figure 4.6: **Link between LipNet1 and generalization gap, dataset size and cross-entropy temperature.** We train a CNN on different fractions of the CIFAR10 train set (2%, 5%, 10%, 25%, 50% and 100% on x-axis) with different values of temperature τ (highlighted by different colors). Train (resp. validation) accuracy forms the upper (resp. lower) bound of each envelope. As τ increases, more samples are required to reduce the generalization gap. Conversely, training a LipNet1 network with small τ is equivalent to training a Lipschitz network with small L: the network generalizes well but the accuracy reaches a plateau (under-fitting). The AllNet network (in red) severely overfit: the generalization gap is large and validation accuracy corresponds to the limit that would reach a LipNet1 as τ increases.

Proposition 8. Optimizing BCE over AllNet leads to divergence. *Let f_t be a sequence of neural networks, that minimizes the BCE over a non-trivial training set (at least two different examples with different labels) of size p, i.e assume that:*

$$\lim_{t\to\infty} \frac{1}{p}\sum_{i=1}^{p} \mathcal{L}_\tau(f_t(x_i), y_i) = 0. \tag{4.21}$$

Let L_t be the Lipschitz constant of f_t. Then $\lim_{t\to\infty} L_t = +\infty$. There is at least one weight matrix W such that $\lim_{t\to\infty} \|W_t\| = +\infty$. Furthermore, the predicted probabilities are saturated:

$$\lim_{t\to\infty} \sigma(f_t(x_i)) \in \{0, 1\}. \tag{4.22}$$

Proof. This result only requires to take a look at the logits of two examples having different labels. Let $t \in \mathbb{N}$. For the pair i, j, as $y_i \neq y_j$, by positivity of $\mathcal{L}$ we must have:

$$0 \leq \mathcal{L}(f_t(x_i), +1) + \mathcal{L}(f_t(x_j), -1) \leq \mathcal{E}(f_t, X). \tag{4.23}$$

As the right hand side has limit zero, we have:

$$\begin{aligned} &\lim_{t\to\infty} \mathcal{L}(f_t(x_i), +1) = \lim_{t\to\infty} \mathcal{L}(f_t(x_j), -1) = 0 \\ \implies &\lim_{t\to\infty} -f_t(x_i) = \lim_{t\to\infty} f_t(x_j) = -\infty. \end{aligned} \tag{4.24}$$

Consequently $\lim_{t\to\infty} |f_t(x_i) - f_t(x_j)| = +\infty$. By definition $L_t \geq \frac{|f_t(x_i) - f_t(x_j)|}{\|x_i - x_j\|}$ so $\lim_{t\to\infty} L^t = +\infty$. □

This issue is especially important since Lipschitz constant and adversarial vulnerabilities are related (Nar et al., 2019). Indeed, the existence of the adversarial attack itself is the proof that the local Lipschitz constant is high. The predicted probability $\sigma(f(x))$ will either be 0 or 1 (regardless of the train set), which do not carry any useful information on the true confidence of the classifier, especially in the *out-of-distribution* setting.

Exemple 4.5. divergence of the optimization

Even on toy example 6 with a trivial model, the minimization problem is ill-defined. Without weight regularization, the minimizer can not be attained. This is compliant with the high Lipschitz constant of AllNet networks that have been observed in practice Scaman and Virmaux (2018), and is confirmed by our experiment on MNIST with a ConvNet (see Figure 4.7).
The behavior of example 6 can be observed at larger scale on MNIST with a Convolutional neural network. We used 3×3 convolution filters of widths $32 \to 64$ with **MaxPool** and **ReLU**, followed by a flattening operation and densely connected layers of widths $256 \to 10$. Newton's method cannot be used due to its memory requirements on ConvNet. We tested SGD with learning rate $\eta = 0.1$ and momentum $m = 0.9$, and Adam with learning rate $\eta = 1e-3$ and other default parameters. Experiments were run both in *float32* and *float64* precision. We monitor the maximum spectral norm of the weights of the network throughout training for each epoch $t \in \mathbb{N}$:

$$\mathcal{M}^t = \max_i \|W_i^t\|_2.$$

We report $\mathcal{M}^t$ as function of epoch t in Figure 4.7. The validation accuracy is above 98% after the first epoch, and fluctuates between 98.5% and 99.5% during the following epochs (in either cases). Similarly the validation loss fluctuates between $1e-1$ and 10^{-3}. We see that on this simple task the spectral norm of weight matrices is multiplied by 5 over the course of 25 epochs, , whereas the validation accuracy remains the same after the first epoch (around 99%). Interestingly, on this experiment the vanishing gradient phenomenon cannot be observed after 25 epochs and the results are robust with respect to the precision of the floating point arithmetic.

This is compliant with the observations made in the literature about the high Lipschitz constant of AllNet networks (Scaman and Virmaux, 2018). We observe that Adam makes the problem worse, even if its learning rate is smaller. This may explain why many practitioners reported that Adam was more susceptible to overfit than SGD with a carefully tuned learning rate scheduling.

We can always find a network reaching arbitrary small loss on the train set, and arbitrary high loss on the test set. Hence, for AllNet networks increasing the size of the training set does not give any formal guarantee to generalization capabilities in general.

Proposition 9 (AllNet networks can always overfit)**.** *Assume that distributions P and Q admit a pdf. Let $n \in \mathbb{N}$, $M > 0$ and $\epsilon > 0$. Let $(x_i, y_i)_{1 \leq i \leq p}$ be a sample of p iid random variables with law $\mathbb{P}_{XY}$ with $x_i \neq x_j$ for all $i \neq j$. Then there exists $f^* \in AllNet$ such that:*

$$f^* \in \{f \in AllNet \,|\mathcal{E}_p(f) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_T(f(x_i), y_i) \leq \epsilon\}$$

and

$$\mathcal{E}_\infty(f^*) = \mathbb{E}_{(x,y)\sim\mathbb{P}_{XY}}[\mathcal{L}_T(f^*(x), y)] \geq M.$$

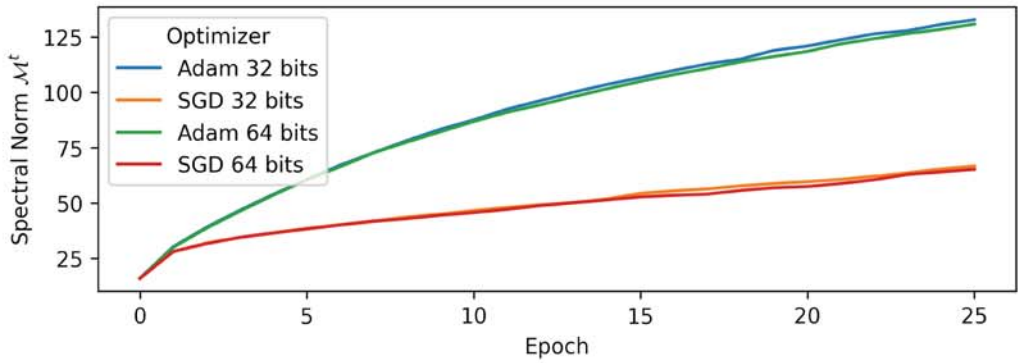
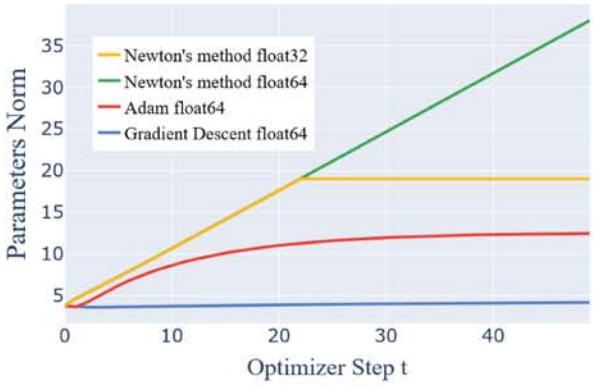


Figure 4.7: **Maximum spectral norm of the weights of a simple ConvNet of AllNet trained with different optimizers on MNIST dataset**. The validation accuracy remains above 98.5% after the second epoch but the network's weights do not converge: the spectral norm seems to grow indefinitely.

Proof. The proof follows the strategy of Proposition 8. Let $d = \min_{\substack{1\leq i,j\leq n \\ i\neq j}} \|x_i - x_j\|$ the minimum distance between dataset points. We extend the dataset with a new point $(x_{n+1}, y = 1))$ chosen such that $\|x_j - x_{n+1}\| \geq \frac{\delta}{2}$ for all $1 \leq j \leq n$. Then we transform this collections of $n+1$ Diracs functions $\sum_{i=1}^{n+1} \frac{1}{n+1}\delta_{x_i}$ into a a distribution P that admits a pdf by replacing each Dirac with the uniform distribution over the ball of radius $r = \frac{d}{6}$ which yields $P = \sum_{i=1}^{n+1} \frac{1}{n+1}\mathbb{U}(\mathfrak{B}(x_i, r))$. All the balls are disjoint so it exists $f \in$ AllNet such that $\operatorname{sign} f(x_i) = y_i$ for all $1 \leq i \leq n$ and $\operatorname{sign} f(x_{n+1}) = -1$. Now let $|f(x_i)| \to \infty$ to guarantee that $\mathcal{E}_p(f) \to 0$ and $\mathcal{E}_\infty(f) \to \infty$. □

Exemple 4.6. Linear classifier

Consider a classification task on $\mathbb{R}$ with linearly separable inputs $\{-1, 1\}$ and labels $\{-1, 1\}$. We use an affine model $f(x) = Wx + b$ for the logits (with $W \in \mathbb{R}$ and $b \in \mathbb{R}$) (one-layer neural network). It exists $\bar{W}, \bar{b}$ such that f achieves 100% accuracy. However, as noticed in Bishop (2006) (Section 4.3.2) the BCE loss *will not* be zero. The minimization occurs only with the diverging sequence of parameters $(\lambda\bar{W}, \lambda\bar{b})$ as $\lambda \to \infty$. It turns out the infimum is not a minimum!



Fortunately, as soon as the deep learning practitioner restricts itself to a subset of architectures of bounded size, the Proposition 9 is no longer relevant. However, this theorem suggests that if one wants to benefit from useful generalization guarantees, one must keep the architecture of the network fixed once for all while increasing the training set size. This contradicts the trend in deep learning community to use bigger and bigger models when more data becomes available (Resnet-152, GPT-3, etc.). In the light of this observation, the existence of adversarial attacks should be an expected phenomenon.

Lipschitz networks, on the other side, benefit from Proposition 7: minimization of train loss

implies minimization of test loss. Conversely, if the test loss is high and the sample size huge, it means that the train loss is high too.

Furthermore, there is an issue of vanishing gradients with BCE : first order methods struggle to saturate the logits of AllNet networks, whereas second order methods in *float64* diverge as expected. The poor properties of the optimizer, and the rounding errors in 32 bits floating point arithmetic, have greatly contributed to the caveat of BCE minimization remaining mostly unnoticed by the community.

4.3.3 Lipschitz classifiers are PAC learnable

Hinge loss $\mathcal{L}_m^H$ and HKR loss $\mathcal{L}^{hkr}$ benefit from Proposition 7. The certificate $|f(x)|$ can be understood as confidence. Hence, we are interested in a classifier that makes a decision only if the prediction is above some threshold $m > 0$, while $|f(x)| < m$ can be understood as examples x for which the classifier is unsure: the label may be flipped using attacks of norm $\epsilon \leq m$.

In this setting, we fall back to PAC learnability (Valiant, 1984): this theory gives bounds on the number of train samples required to guarantee that the test error will fall below some threshold $0 \leq e < \frac{1}{2}$ with probability at least $1 > \beta \geq 0$, through the use of Vapnik Chervonenkis (VC) dimension bounds (Vapnik and Chervonenkis, 1971).

Remark 4.1. Classical learning theory: a crash course.

We recall below the definition of the Vapnik-Chervonenkis dimension (Vapnik and Chervonenkis, 1971) of a class of hypothesis, that build upon shattered sets.

Definition 12 (**Set shattered by an hypothesis class**). *Let $\mathcal{Y} = \{-1, +1\}$. Let $\mathcal{H}$ be a class of hypothesis - that is, a set of functions $\mathcal{X} \to \mathcal{Y}$. The set of points $(x_i)_{1 \leq i \leq N} \in \mathcal{X}^N$ is said to be **shattered** by $\mathcal{H}$ if for every sequence of labels $(y_i)_{1 \leq i \leq N} \in \mathcal{Y}^N$, there exists an hypothesis $h \in \mathcal{H}$ such that for every $1 \leq i \leq N$ we have $h(x_i) = y_i$.*

Definition 13 (**Vapnik-Chervonenkis dimension**). *The VC dimension of $\mathcal{H}$, denoted $VC_{dim}(\mathcal{H})$, is the greatest integer $N \in \mathbb{N}$ such that it exists a sequence of points $(x_i)_{1 \leq i \leq N} \in \mathcal{X}^N$ shattered by $\mathcal{H}$.*

Roughly speaking, the VC dimension of $\mathcal{H}$ is the size of the biggest set of points such that $\mathcal{H}$ agrees with any label assignment on this set of points. It measures the capacity of a set of classifiers $\mathcal{H}$ to separate some sets of points. The interest of VC dimension introduced in Definition 13 is its link with Probably Approximately Correct (PAC) learning (Valiant, 1984).

Definition 14 (**Agnostic Probably Approximately Correct (PAC) learnability**). *An hypothesis class $\mathcal{H}$ of functions $\mathcal{X} \to \mathcal{Y}$ is PAC learnable if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm $\mathcal{D} \mapsto h_m$ such that for every $(e, \beta) \in (0,1)^2$, for any distribution $\mathbb{P}_{XY}$ on $\mathcal{X} \times \mathcal{Y}$, for any dataset $\mathcal{D} = ((x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)) \overset{iid}{\sim} \mathbb{P}_{XY}$ of size $m \geq m_{\mathcal{H}}(e, \beta)$, we have:*

$$\mathbb{P}(E_{\mathbb{P}_{XY}}(h_m) \leq \min_{h \in \mathcal{H}} E_{\mathbb{P}_{XY}}(h) + e) \geq 1 - \beta.$$

We denote by $E_{\mathbb{P}_{XY}}(h) := \mathbb{E}_{(x,y) \sim \mathbb{P}_{XY}}[\mathbb{1}\{h(x) \neq y\}]$ the empirical risk: the expectation of error function over $\mathbb{P}_{XY}$.

Roughly speaking, for an agnostic PAC learnable class, the probability to pick the best hypothesis $h^* \in \mathcal{H}$ up to error $e > 0$ happens with probability at least $1 - \beta > 0$ over

datasets of size at least $m_{\mathcal{H}}(e, \beta)$ sampled from distribution $\mathbb{P}_{XY}$. This definition captures the hypothesis classes that are "small enough" such that a reasonably high number of samples allows you to pick the best hypothesis by high probability.

The implication "finite VC dimension" $\implies$ "agnostic PAC learnable" is a classical result from Blumer et al. (1989). This motivates to compute the VC dimension of Lipschitz classifiers: it yields PAC learnability results.

Proposition 10. 1-Lipschitz Functions with margin are PAC learnable. *Assume P and Q have bounded support $\mathcal{X}$. Let $m > 0$ the margin. Let $\mathcal{C}^m(\mathcal{X}) = \{c_f^m : \mathcal{X} \to \{-1, \perp, +1\}, f \in Lip_1(\mathcal{X}, \mathbb{R})\}$ be the hypothesis class defined as follow.*

$$c_f^m(x) = \begin{cases} +1 & \text{if } f(x) \geq m, \\ -1 & \text{if } f(x) \leq -m, \\ \perp & \text{otherwise, meaning "}f \text{ doesn't feel confident".} \end{cases} \tag{4.25}$$

Let $\mathfrak{B}$ be the unit ball. Then the VC dimension of $\mathcal{C}^m$ is finite:

$$(\frac{1}{m})^n \frac{vol(\mathcal{X})}{vol(\mathfrak{B})} \leq VC_{dim}(\mathcal{C}^m(\mathcal{X})) \leq (\frac{3}{m})^n \frac{vol(\mathcal{X})}{vol(\mathfrak{B})}. \tag{4.26}$$

Proof. This approach with margins m yields objects known in the literature as m-fat shattering sets Gottlieb et al. (2014).

The VC dimension of $\mathcal{C}^m(\mathcal{X})$ is the maximum size of a set shattered by $\mathcal{C}^m(\mathcal{X})$. As the functions f are 1-Lipschitz, if $c_f^m(x) = -c_f^m(y)$ then $f(x) \geq m$, $f(y) \leq m$ and $\|x - y\| \geq 2m$. Consequently, a finite set $X \subset \mathcal{X}^n$ is shattered by $\mathcal{C}^m(\mathcal{X})$ if and only if for all $x, y \in X$ we have $\mathfrak{B}(x, m) \cap \mathfrak{B}(y, m) = \varnothing$ where $\mathfrak{B}(x, m)$ is the open ball of center x and radius m.

The maximum number of disjoint balls of radius m that fit inside $\mathcal{X}$ is known as the **packing number** of $\mathcal{X}$ with radius m. $\mathcal{X}$ is bounded, hence its packing number is finite.

The bounds on the packing number are a direct application of Szarek (1998) (Lemma 1). □

Interestingly if the classes are ϵ separable ($\epsilon > 0$), choosing $m = \epsilon$ guarantees that 100% accuracy is reachable. Prior over the separability of the input space is turned into VC bounds over the space of hypothesis. When $m = 0$ the VC dimension of space $\mathcal{C}^m(\mathcal{X})$ becomes infinite and the class is not PAC learnable anymore: the training error will not converge to test error in general, regardless of the size of the training set. It is not a contradiction with Proposition 7: error $E(c_f^m(x))$ lacks continuity w.r.t $f(x)$ so it is not a consistent estimator. Note that this result is compliant with earlier observations of Bartlett (1996).

Warning 4.2. Architecture independant bound

This VC bound is *architecture independent* which contrasts with the rest of literature on AllNet networks. Practically, it means that the LipNet1 network architecture can be chosen as big as we want without risking overfitting, as long as the margin m is chosen appropriately.

Proposition 11 also provides an architecture dependant bound for LipNet1 networks. With GroupSort2 activation function, we get the following rough upper bound.

Proposition 11. VC dimension of LipNet1 neural networks. *Let $f_\theta : \mathbb{R}^n \to \mathbb{R}$ a LipNet1 neural network with parameters $\theta \in \Theta$, with **GroupSort2** activation functions, and a total of W neurons. Let $\mathcal{H} = \{sign f_\theta | \theta \in \Theta\}$ the hypothesis class spanned by this architecture. Then we have:*

$$VC_{dim}(\mathcal{H}) = \mathcal{O}\left((n+1)2^W\right). \tag{4.27}$$

From Proposition 11 we can derive generalization bounds using PAC theory. Note that most results on VC dimension of neural network use the hypothesis that the activation function is applied element-wise (such as in Bartlett et al. (2019)) and get asymptotically tighter lower bounds for ReLU case. This hypothesis does not apply to GroupSort2 which is known to be more expressive (Tanielian and Biau, 2021), however we believe that this preliminary result can be strengthened.

Our result is actually a bit more general and applies more broadly to activation functions that piece-wise linear and partition the input space into convex sets.

Proof. The proof uses the number of affine pieces generated by GroupSort2 activation function, and the VC dimension of piecewise affine classifiers with convex regions.

First, we need the following lemma.

Lemma 1 (Piecewise affine function)**.** *Let $\mathcal{H}$ a class of classifiers that are piecewise affine, such that the pieces form a convex partition of $\mathbb{R}^n$ with B pieces (each piece of the partition is a convex set). Then we have:*

$$VC_{dim}(\mathcal{H}) = \mathcal{O}\left((n+1)B^2\right).$$

The proof of Lemma 1 is detailed below.

Let $\mathcal{G}(N)$ be the **growth function** (Vapnik, 2013) of $\mathcal{H}$. According to Sauer's lemma (Vapnik, 2013) if it grows polynomially with the number of points, then the degree of the polynomial is an upper bound on the VC dimension. We will show that is indeed the case by computing a crude upper bound of the degree. Assume that we are given N points, and N big enough such that Sauer's lemma can be applied.

Assume that we can choose freely the convex partition, and then only the affine classifier inside each piece. In general for neural networks that might not be the case (the boundary between partitions depends of the affine functions inside it, since neural networks are continuous); however, we are only interested in an upper bound so we can consider this generalization.

Each piece of the partition is a polytope (León and Ziegler, 2018). Each polytope is characterized by a set of exactly $B-1$ affine inequalities since each polytope is the intersection of $B-1$ halfspaces (León and Ziegler, 2018). The whole partition is characterized by $\frac{B(B-1)}{2}$ affine inequalities. We divide by two because of the symmetry. Hence there exists an injective map from the set of convex partitions with B pieces into $(\mathbb{R}^{n+1})^{\frac{B(B-1)}{2}}$. It is not a bijective map in general, since different systems might describe the same partition, and some degenerate systems do not correspond to partitions at all.

We split the problem and consider each one of the $\frac{B(B-1)}{2}$ inequalities independently. According to Sauer's lemma, there is $\mathcal{O}(N^{n+1})$ ways to place the first hyperplane characterizing the first halfspace. Idem for the second hyperplane, and so on. Hence, there is at most $\mathcal{O}((N^{n+1})^{\frac{B(B-1)}{2}})$ ways to assign the N points to the B convex bodies.

Each convex body (among the B of them) contains atmost N points, on which (still according to Sauer's lemma) there is at most $\mathcal{O}(N^{n+1})$ way to assign them labels $+1$ or -1, since the classifier is piecewise **affine**.

Consequently, we have $\mathcal{G}(N) = \mathcal{O}((N^{n+1})^{\frac{B(B-1)}{2}}(N^{n+1})^B) = \mathcal{O}((N^{n+1})^{\frac{B(B+1)}{2}}) = \mathcal{O}((N^{n+1})^{B^2})$. Sauer's lemma allows us to conclude:

$$VC_{\text{dim}}(\mathcal{H}) = \mathcal{O}\left((n+1)B^2\right).$$

Proof of the main result. Now, we need to prove that f is piecewise affine and the number of such pieces is not greater than $\prod_{i=1}^{k} 2^{\frac{w_i}{2}} = \sqrt{2^W}$, where w_i is the number of neurons in layer i. We proceed by induction on the depth of the neural network. For depth $K = 0$ we have an affine function $\mathbb{R}^n \to \mathbb{R}$ which contains only one affine piece by definition (the whole domain), so the result is true.

Now assume that a neural network $\mathbb{R}^{w_1} \to \mathbb{R}$ of depth K with widths $w_2 w_3 \dots w_k$ has S_k affine pieces. The enumeration starting at w_2 is not a mistake: we pursue the induction for a neural network $\mathbb{R}^n \to \mathbb{R}$ of depth $K+1$ and widths $w_1 w_2 \dots w_k$. The composition of affine function is affine, hence applying an affine transformation $\mathbb{R}^n \to \mathbb{R}^{w_1}$ preserves the number of pieces. The analysis falls back to the number of distinct affine pieces created by GroupSort2 activation function. If such activation function creates S pieces then we have the immediate bound $S_{K+1} \leq S S_k$.

Let $(Jf)(x) \in \mathbb{R}^{w_1 \times w_1}$ be the Jacobian of the GroupSort2 operation evaluated in x. The cardinal $|\{(Jf)(x), x \in \mathbb{R}^{w_1}\}|$ is the number of distinct affine pieces. For GroupSort2 we have combinations of $\frac{w_i}{2}$ MinMax gates. Each MinMax gate is defined on $\mathbb{R}^2$ and contains two pieces: one on which the gate behaves like identity and the other one on which the gate behaves like a transposition. Consequently we have $S_{k+1} \leq 2^{\frac{w_k}{2}} S_k$ and unrolling the recurrence yields the desired result.

Finally, we just need to apply the Lemma 1 with $B = \sqrt{2^W}$.

□

4.4 Calibration of temperature

The temperature parameter of cross-entropy enjoys a nice interpretation as the strength of an entropic regularization term. Like any regularization term, it can be optimized on a validation set. Here, it has the advantage of being understood as a form of calibration, when the noise on the train set and the set are different. The preliminary work of this section is unpublished.

4.4.1 Regularized predictions functions with entropy penalty

This section takes strong inspiration from the works of Blondel et al. (2019) and Blondel et al. (2020). We recall below the main tools introduced in these papers. In the following, we consider a supervised classification task with dataset $\mathcal{D}$ consisting of pairs (x_i, y_i) with $x_i \in \mathcal{X}$ the input data, and $y \in \mathcal{Y} = [\![1, \dots, K]\!]$ the label, where K is the number of classes. We consider a parametrized model $f_\theta : \mathcal{X} \to \mathcal{S}$ producing a score vector $s := f_\theta(x) \in \mathcal{S}$ living in score space $\mathcal{S} \subseteq \mathbb{R}^d$. Typically $\mathcal{S} = \mathbb{R}^K$ and the score is used to predict a label $\boldsymbol{f}$ having the maximum score:

$$\boldsymbol{f}(s) := \underset{i \in [\![1,\dots,K]\!]}{\arg\max}\, s_i. \tag{4.28}$$

This approach can be generalized to any convex score space $\mathcal{S}$:

$$\boldsymbol{f}(s) := \underset{y \in \mathcal{S}}{\arg\max} \langle s, y \rangle. \tag{4.29}$$

The inner product $\langle s, y \rangle$ measures the affinity between the input x and the prediction y, through the score predictor f_θ. We see that Equation 4.28 is actually a particular case of Equation 4.29 when the score space § is restricted to the canonical basis $\mathcal{S} = \{e_i | 1 \leq i \leq n\}$.

Definition 15 (Regularized prediction function)**.** *We consider the convex-hull of the scores conv*$(\mathcal{S}) := \{\mathbb{E}_p[Y], p \in \Delta^{|\mathcal{S}|}\}$ *where* $\Delta^{|\mathcal{S}|}$ *is the probability simplex in dimension* $|\mathcal{S}|$*. Furthermore we add a regularization term* $\Omega : \Delta^{|\mathcal{S}|} \to \mathbb{R}$ *to obtain the final formulation studied in:*

$$\boldsymbol{f}_\Omega(s) \in \underset{\mu \in conv(\mathcal{S})}{\arg\max} \langle s, \mu \rangle - \Omega(\mu). \tag{4.30}$$

Importantly, μ is a regularization *of the prediction*, not of the parameters W. This framework is very general and allows us to tackle the case of Softmax for example, as detailed in Blondel et al. (2019).

Exemple 4.7. Softmax as regularized prediction function (Blondel et al., 2019)

Consider $\Omega(s) = -H(s) + \mathbb{1}\{s \in \Delta\}$ where $\mathbb{1}\{s \in \Delta\}$ is indicator function of the probability simplex of dimension K:

$$\mathbb{1}\{s \in \Delta\} := \begin{cases} 0 \text{ if } s_i \leq 0 \text{ and } \sum_{i=1}^K s_i = 1, \\ +\infty \text{ otherwise,} \end{cases} \tag{4.31}$$

and $H(s) = -\sum_{i=1}^K p_i \log p_i$ the celebrated Shannon entropy. Then it can be shown that:

$$\boldsymbol{f}_\Omega(s) = \text{softmax}(s) = \frac{\exp s}{\sum_{i=1}^K \exp s_i}. \tag{4.32}$$

The regularization Ω can be used to define a special kind of loss, the Fenchel-Young loss.

Definition 16 (Fenchel-Young losses, Blondel et al. (2019))**.** *The Fenchel-Young loss $L_\Omega : \mathbb{R}^d \times dom(\Omega) \to \mathbb{R}_+$ generated by Ω is defined as:*

$$L_\Omega(s, y) := \Omega^*(s) + \Omega(y) - \langle s, y \rangle, \tag{4.33}$$

where

$$\Omega^*(s) := \sup_{\mu \in dom(\Omega)} \langle s, \mu \rangle - \Omega(\mu) \tag{4.34}$$

is the Fenchel conjugate of Ω. Notably we have:

$$\begin{aligned} L_\Omega(s, y) &= \Omega(y) - \langle s, y \rangle - \Omega(\boldsymbol{f}_\Omega(s)) + \langle s, \boldsymbol{f}_\Omega(s) \rangle \\ &= (\Omega(y) - \Omega(\boldsymbol{f}_\Omega(s))) - \langle s, y - \boldsymbol{f}_\Omega(s) \rangle. \end{aligned} \tag{4.35}$$

Furthermore the following holds:

$$\boldsymbol{f}_\Omega(s) \in \underset{\mu \in dom(\Omega)}{\arg\min}\ L_\Omega(s, \mu). \tag{4.36}$$

In the case of Softmax, we fall back to the familiar Softmax-Crossentropy loss, as derived in Boyd and Vandenberghe (2004) and (Blondel et al., 2019).

Exemple 4.8. Fenchel-Young loss from entropic regularization

Assume that $\Omega_{CE}(s) = -H(s) + \mathbb{1}\{s \in \Delta\}$. Then:

$$L_{\Omega_{CE}}(s, y) = -s_k + \log \sum_{i=1}^K \exp s_i \tag{4.37}$$

where the ground truth y is the one-hot encoding of the class k.

Lipschitz network training with entropic regularization

In this section, we apply the previous tools to train LipNet1 with Cross-entropy and temperature scaling τ. In the following we denote it $L^\tau_{CE} := L_{\Omega_{CE}}(\tau s, y) = L_{\Omega_{CE}}(\tau s, y)$. We assume that the dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ is sampled from the joint distribution $\mathcal{P}^{\otimes n}$. For a score functions $f : \mathcal{X} \to \mathcal{S}$ we define the population risk (i.e the "test loss") as follow:

$$\mathcal{E}(f) := \mathbb{E}_{(x,y)\sim\mathcal{P}}[L^\tau_{CE}(f(x), y)]. \tag{4.38}$$

Let $\mathcal{F} \subseteq (\mathcal{X} \to \mathcal{S})$ be a set of score functions over a suitable domain. We consider the classical setting of Empirical Risk Minimization (ERM) where the expectation is taken over the *train set* $\mathcal{D}$:

$$\mathcal{M}(\mathcal{F}, L, \mathcal{D}) = \arg\inf_{f\in\mathcal{F}} \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(f(x), y)], \tag{4.39}$$

where $\mathcal{M}(\mathcal{F}, L) \subset \mathcal{F}$ is the set of minimizers in function space $\mathcal{F}$ and loss $\mathbb{R}^d \times \mathrm{dom}(\Omega) \to \mathbb{R}_+$. Then we have:

$$\begin{aligned}
\mathcal{M}(\mathrm{Lip}_\tau(\mathcal{X}, \mathcal{S}, \mathcal{D}), L_{\Omega_{CE}}) &= \mathcal{M}(\tau\mathrm{Lip}_1(\mathcal{X}, \mathcal{S}), L_{\Omega_{CE}}) \\
&= \arg\inf_{f\in\mathrm{Lip}_1(\mathcal{X},\mathcal{S})} \mathbb{E}_{(x,y)\sim\mathcal{D}}[L_{\Omega_{CE}}(\tau f(x), y)] \\
&= \arg\inf_{f\in\mathrm{Lip}_1(\mathcal{X},\mathcal{S})} \mathbb{E}_{(x,y)\sim\mathcal{D}}[L^\tau_{CE}(f(x), y)].
\end{aligned} \tag{4.40}$$

Now, take a closer look at $L^\tau_{CE}(f(x), y)$. Per the *temperature scaling property* (proposition 2 item 5 of Blondel et al. (2020)) we have:

$$\begin{aligned}
L^\tau_{CE}(f(x), y) &= L_{\Omega_{CE}}(\tau f(x), y) \\
&= \tau L_{\Omega_{CE}/\tau}(f(x), y) \\
&= \tau\left((\Omega(y) - \Omega(\boldsymbol{f}_{\Omega/\tau}(f(x)))/\tau) - \langle f(x), y - \boldsymbol{f}_{\Omega/\tau}(f(x))\rangle\right).
\end{aligned} \tag{4.41}$$

Here, τ appears as a common pre-factor that does not depend on (x, y), so it does not change $\arg\inf_{f\in\mathrm{Lip}_1(\mathcal{X},\mathcal{S})}$ and it can be dropped safely. If we assume that y is a deterministic label (as often in supervised learning), we have $\Omega(y) = 0$. We end-up with:

$$-\Omega(\boldsymbol{f}_{\tau\Omega}(f(x)))/\tau - \langle f(x), y - \boldsymbol{f}_{\Omega/\tau}(f(x))\rangle. \tag{4.42}$$

We recall that:

$$\begin{aligned}
\boldsymbol{f}_{\tau\Omega}(f(x)) &= \arg\max_{\mu\in\Delta^{|\mathcal{S}|}} \langle f(x), \mu\rangle - \Omega(\mu)/\tau \\
&= \arg\max_{\mu\in\Delta^{|\mathcal{S}|}} \langle \tau f(x), \mu\rangle - \Omega(\mu) \\
&= \mathrm{softmax}(\tau f(x)) \in \Delta^K.
\end{aligned} \tag{4.43}$$

We see that the temperature parameter $\beta := 1/\tau$ plays the role of entropic regularization in the predictions. Finally, $\mathcal{M}(\mathrm{Lip}_\tau(\mathcal{X}, \mathcal{S}), L_{\Omega_{CE}})$ expands as:

$$\tau \cdot \arg\operatorname{supp}_{f\in\mathrm{Lip}_1(\mathcal{X},\mathcal{S})} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\beta\Omega(\boldsymbol{f}_{\beta\Omega}(f(x))) + \langle f(x), y - \boldsymbol{f}_{\beta\Omega}(f(x))\rangle]. \tag{4.44}$$

For the population risk, abusing the notation $\mathbb{E}_{(x,y)\sim\mathcal{P}}[\cdot] = \mathcal{P}[\cdot]$, we obtain:

$$f^\beta \in \arg\operatorname{supp}_{f\in\mathrm{Lip}_1(\mathcal{X},\mathcal{S})} \underbrace{\mathcal{P}[\beta\Omega(\boldsymbol{f}_{\beta\Omega}(f(x))) - \langle f(x), \boldsymbol{f}_{\beta\Omega}(f(x))\rangle]}_{\text{Regularization dependent}} + \underbrace{\mathcal{P}[\langle f(x), y\rangle]}_{\text{Label dependant.}}. \tag{4.45}$$

Remark 4.2. Asymptotic regimes.

We recognize the asymptotic behavior of cross-entropy loss.

- **Case of $\beta \to 0$.** The regularization term vanishes. Therefore $\boldsymbol{f}_{\beta\Omega}(f(x)) = e_k$ with $k = \arg\max_i s_i$ the argmax prediction. Finally, only remains the term $\langle f(x), y - e_k \rangle$ with $y - e_k$ which is either $\mathbf{0}$, either a vector full of zero's with two coordinates $+1$ and -1 respectively. This is the familiar *0-1 loss*, equal to the error, piecewise constant, not differentiable everywhere.
- **Case of $\beta \to +\infty$.** The regularization term is dominant. As a consequence $\boldsymbol{f}_{\beta\Omega}(f(x)) = (1/n)\mathbf{1}$. The contribution of the ground truth y vanishes and the optimum degenerates to the constant classifier of maximum entropy.

We may be interested in the dynamic of the sequence of 1-Lipschitz functions f^β when $\tau \to +\infty$, since it mimics the dynamic of the training of an unconstrained network, up to the rescaling factor τ. We will see in the next section that this regularization allows for a nice interpretation as a train-test distribution shift, whether noise in the measurements x or label noise y.

4.4.2 Tuning of the temperature on a calibration set

Now that the role of (inverse) temperature τ is understood as (inverse) entropic regularization of the predictions, a natural question to ask is whether or not it can improve accuracy when noise corrupts the data. To test this hypothesis, we minimize the risk on train set $\mathcal{D}$, but we measure the risk on a *calibration* set $\mathcal{T}$. Importantly, we do *not* assume that $\mathcal{D}$ and $\mathcal{T}$ are finite samples from the same distribution, to take into account distribution shifts.

We propose algorithm 4. In this setting, the optimization is performed on the set of 1-Lipschitz functions by minimizing the loss on the train set $\mathcal{D}$, and then by minimizing the loss on the set calibration set $\mathcal{T}$ solely with optimization of temperature β. This algorithm shares a similar spirit with the "stability control loop" studied in the paper Gupta et al. (2022a), where the Lipschitz constant is updated periodically.

Algorithm 4 Self calibration of Cross-entropy temperature

Input: 1-Lipschitz neural network architecture f^β, initial temperature β_0
Input: Train set $\mathcal{D}$, calibration set $\mathcal{T}$

1: **repeat**
2: Fit the empirical risk minimizer of Cross-Entropy with fixed temperature β_t:

$$f_{t+1} \in \operatorname*{arg\,inf}_{f \in \mathrm{Lip}_1(\mathcal{X},\mathcal{S})} \mathbb{E}_{(x,y)\sim\mathcal{D}}[L_{CE}^{\beta_t}(f(x), y)].$$

3: Fit the temperature β_t of Cross-Entropy with fixed predictor f_t:

$$\beta_{t+1} \in \operatorname*{arg\,inf}_{\beta \in \mathbb{R}_+} \mathbb{E}_{(x,y)\sim\mathcal{T}}[L_{CE}^{\beta_t}(f(x), y)].$$

4: **until** convergence of β_T.

Exemple 4.9. Temperature calibration on Two Moons dataset.

In this setting we consider the Two Moons dataset in three different scenarios.

1. **Clean-Clean** scenariio. Here $\mathcal{D}$ and $\mathcal{T}$ are sampled from the same distribution P, with very small measurement errors in x position. The two moons supports are disjoint.
2. **Noisy-Noisy** scenariio. Here $\mathcal{D}$ and $\mathcal{T}$ are sampled from the same distribution Q, with huge measurement errors in x position. The two moons strongly overlap.
3. **Clean-Noisy** scenariio. Here $\mathcal{D}$ is sampled from P with small measurement errors, but $\mathcal{T}$ is sampled from Q, with huge measurement errors.

Settings 1 and 2 fall under the ERM paradigm, which has already been handled in section 4.3. They serve as a "sanity check". What matters the most is setting 3. In this last setting, the calibration set $\mathcal{T}$ is intended as representative of the (noisy) testing data, and may be scarce. This third setting is divided into two experiments: in the "control" experiment, conventional training is applied on $\mathcal{D}$ with fixed temperature β, and the loss is computed on the calibration set $\mathcal{T}$. In the "positive" experiment, we apply algorithm 4.

We optimize both neural networks weights θ and temperature parameter β with Adam. The training is stopped once the value of β stabilizes. The results are given in Figure 4.8 and the table below:

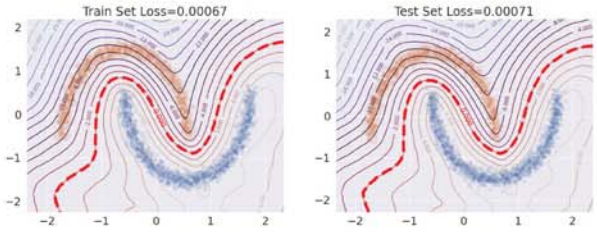
Scenario	Loss on train set $\mathcal{D}$	Loss on calibration set $\mathcal{T}$	Comment
Clean-Clean	0.067×10^{-2}	0.067×10^{-2}	ERM, low loss.
Noisy-Noisy	11×10^{-2}	11×10^{-2}	ERM, higher loss.
Clean-Noisy "control"	0.23×10^{-2}	86×10^{-2}	Distribution shift, high loss.
Clean-Noisy "positive"	14×10^{-2}	42×10^{-2}	Distribution shift, lower loss.

The calibration step on β and $\mathcal{T}$ *changes the whole decision frontier* (better seen on figures), even though the decision frontier parametrized by f is fitted on $\mathcal{D}$. herefore, at test time, the loss is 42×10^{-2} instead of 86×10^{-2}, which is a factor 2 improvement. Of course, this has consequences on the train loss (which is now higher). But assuming that the calibration set $\mathcal{T}$ is representative of "test data", this may improve performance after deployment.

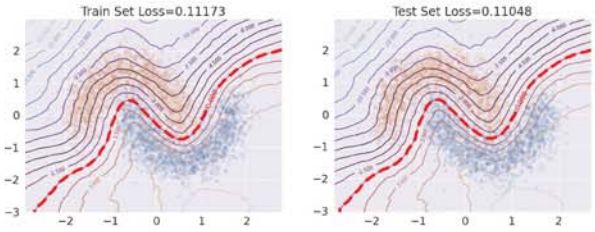
The intricate dynamics of example 9 shade lights on the training of deep conventional networks. Indeed, as shown in Proposition 4, without regularization, the Lipschitz constant of AllNet models tends to blow up. This can be understood as temperature regularization β going to zero, making the Cross-Entropy behave like 0-1 loss. This makes sense: if $\mathcal{D}$ is not considered as a finite sample from some "real data" distribution P, *but as P itself*, this behavior makes sense. But in practice, the real-world data P and $\mathcal{D}$ are not similar. There are distribution shifts, noise, and various effects in play that require regularization. Interestingly, the stochasticity of mini-batch sampling, the stochasticity of dropouts layers (Srivastava et al., 2014), the stochasticity of stochastic depth layers (Huang et al., 2016a): all these sources of randomness and "surprise", plays the role of the calibration set $\mathcal{T}$, *at every new batch* during training.

4.5 Perspectives

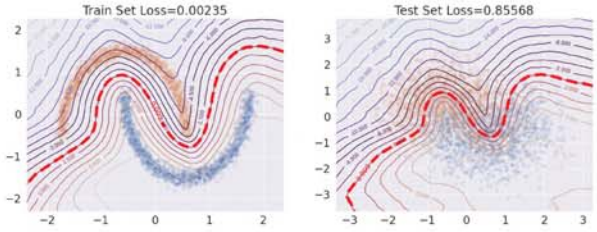
In this section we review the contributions and the possible future works inspired by section 4.4.



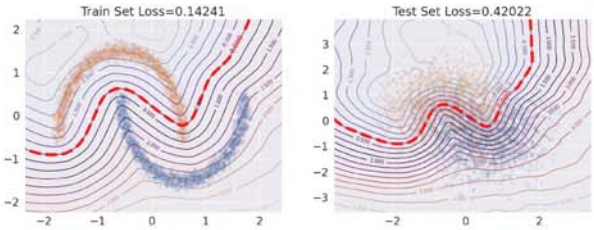
(a) **Clean-Clean** scenario. Fall under ERM paradigm, no noise, train and test losses are equal and small.



(b) **Noisy-Noisy** scenario. Fall under ERM paradigm, high noise, train and test losses are equal and high.



(c) **Clean-Noisy** "control" scenario. Distribution shift paradigm. The classifier trained on $\mathcal{D}$ is over-confided on the noisy $\mathcal{T}$, which induces a high loss.



(d) **Clean-Noisy** "positive" scenario. The classifier is trained on $\mathcal{D}$, but the temperature β is calibrated on $\mathcal{T}$.

Figure 4.8: **Four scenarios with training set $\mathcal{D}$ and calibrating set $\mathcal{T}$.**

Properties	AllNet network	LipNet1 network
Fit any boundary	yes (Hassoun et al., 1995)	yes (Proposition 2)
Robustness certificates	no	yes (Property 1)
Consistent estimator	no	yes (Proposition 7, Figures 4.2, 4.6)
Gradients	exploding or vanishing	preserved for GNP
VC dim. bounds	architecture dependent (Bartlett et al., 2019)	when $m > 0$ (Proposition 10)

Table 4.1: **Summary of notable results and the contributions.**

Properties		AllNet network	LipNet1 network
BCE $\mathcal{L}^{bce}_{\tau}$	minimizer	ill-defined $L_t \to \infty$ (Proposition 8)	attained (Proposition 4)
	remark	vanishing gradient (Ex 6)	L or τ must be tuned (Figure 4.5)
Wasserstein $\mathcal{L}^{W}$	minimizer	ill-defined $L_t \to \infty$	attained, robust (Property 4)
	remark	diverges during training	weak classifier (Proposition 6)
Hinge $\mathcal{L}^{H}_{m}$	minimizer	attained	attained
	remark	no guarantees on margin	m must be tuned
HKR $\mathcal{L}^{hkr}_{m,\alpha}$	minimizer	ill-defined $L_t \to \infty$	accuracy-robustness tradeoff
	remark	diverges during training	α and m must be tuned (Figure 4.5)

Table 4.2: **Summary of losses, influence of the Lipschitz constraint on the optimum**. BCE minimization is ill-posed for AllNet , but vanishing gradient leads to convergence. For margin m small enough, if the classes are separable, 0% training error is achievable by Hinge and HKR.

4.5.1 Future works

Different applications of algorithm 4 can be foreseen such as calibration of predictions (Guo et al., 2017), a way to make the training of Lipschitz constrained network more efficient, and a way to handle distribution shifts even when test data are scarce. When $\mathcal{T}$ shifts away from P, then β can be fitted on $\mathcal{T}$ (which does not require a lot of resources), and then the whole network can be re-retrain on $\mathcal{D}$ (again!) with a different regularization β. Hopefully, this improves generalization. Why would this be possible? Because of the bias-variance tradeoff. Tuning the temperature dynamically is like tuning the bias/variance tradeoff. When $\mathcal{D}$ and $\mathcal{T}$ are close, it is more advantageous to work in a low bias regime (low temperature), whereas when their is a shift it is more important to mitigate variance by increasing temperature.

4.5.2 Conclusion

In this chapter, we challenged the common belief that constraining Lipschitz constant degrades the classification performance of neural networks. We proved that LipNet1 networks exhibit numerous attractive properties (see Table 4.1 in summary): they provide robustness radius certificates without restrictions on their expressive power. They benefit from generalization guarantees. We showed that the hidden parameters of the loss allow to control the generalization gap and certifiable robustness.

While the question of the LipNet1 architecture is often in the spotlight, the loss is overlooked. We pointed out that Cross-Entropy is not necessarily the best choice, margin-based losses, such as hinge or its variant HKR, have appealing properties (table 4.1).

This work aims to be at the intersection between theoretical ML and (empirical) deep learning. Lipschitz constrained networks allow to directly put in perspective mathematical proofs and we are confident that this theory can be verified empirically on very large-scale vision datasets (such as Imagenet (Deng et al., 2009)).

This work also provides a toolbox of results and experiments to serve as a basis for future works. We aim to open new research directions, including outside the field of robust learning. AllNet networks could benefit from LipNet1 literature: the absence of control over the Lipschitz constant of AllNet is mitigated in practice by elements such as mixup or weight decay. Such elements would be better understood by looking at how they affect the (uncontrolled) Lipschitz constant of AllNet .

The efficient training over LipNet1 is still an active research area. Moreover, AllNet networks benefits from architectural elements such as skip connections and batch normalization. As LipNet1 networks get more mature, empirical results will improve, matching theory even more.

Many practices in deep learning entangle the questions of architecture, of generalization, and of optimization. However, these elements usually have unexpected consequences on the nature of the optimum and the optimization process. Our work is a first step toward a better separation of these components and their role.

Chapter 5

Neural signed distance functions and applications to One Class classification

This chapter is mostly adapted from the corresponding publication:

Louis Béthune, Paul Novello, Thibaut Boissin, Guillaume Coiffier, Mathieu Serrurier, Quentin Vincenot, Andres Troya Galvis, **Robust One-Class Classification with signed distance function using 1-Lipschitz neural networks**, International Conference on Machine Learning (ICML), *2023*. See Béthune et al. (2023).

One class classification (OCC) is an instance of binary classification where all the points of the dataset at hand belong to the same (positive) class. The challenge of this task is to construct a decision boundary without using points from the other (negative) class. It has various safety-critical applications in anomaly detection, for instance, to detect banking fraud, cyber-intrusion or industrial defect, in out-of-distribution detection, to prevent wrong decisions of Machine Learning models, or in Open-Set-Recognition. However, OCC algorithms suffer from limitations such as the **lack of negative data**, and **robustness issues** (Azizmalayeri et al., 2022), the latter being an under-explored topic in the OCC spectrum. Even though some algorithms do not use negative examples, many work cope with the lack of negative data with Negative Sampling, either artificially (Sipple, 2020) or using outlier exposure (Han et al., 2022; Fort et al., 2021). However, such samplings are often biased or heuristic. As for robustness, although some works design robust algorithms (Goyal et al., 2020; Lo et al., 2022), it is always only empirically demonstrated (Han et al., 2022). Few works provide theoretical certifications (we only found Bitterwolf et al. (2020) based on interval bounds propagation). In this work, we leverage the properties of 1-Lipschitz networks to provide certifications.

In this chapter, we introduce a new framework to perform OCC based on the Signed Distance Function (SDF), a function traditionally used in computer graphics. Assume the positive samples are independently and identically obtained from a distribution $\mathbb{P}_X$ with compact support $\mathcal{X} \subset \mathbb{R}^d$. Let $\partial\mathcal{X} = \overline{\mathcal{X}}/\mathring{\mathcal{X}}$ be the boundary of the distribution. The Signed Distance Function is the function $\mathcal{S} : \mathbb{R}^d \to \mathbb{R}$:

$$\mathcal{S}(x) = \begin{cases} d(x, \partial\mathcal{X}) & \text{if } x \in \mathcal{X}, \\ -d(x, \partial\mathcal{X}) & \text{otherwise,} \end{cases} \tag{5.1}$$

where $d(x, \partial\mathcal{X}) = \inf_{z \in \partial\mathcal{X}} \|x - z\|_2$. The idea of our algorithm, which we call One Class Signed Distance Function (OCSDF) is to learn the SDF to the boundary of the positive data distribution and use it as a normality score. We show that the Hinge Kantorovich-Rubinstein (HKR) loss introduced by Serrurier et al. (2021) allows provably learning the SDF with a 1-Lipschitz network.

SDF exhibits desirable properties. First, by implicitly parametrizing the domain $\mathcal{X}$, it allows to efficiently sample points outside of $\mathcal{X}$ and to perform principled Negative Sampling. Second, the SDF fulfills the Eikonal equation: $\|\nabla_x \mathcal{S}(x)\| = 1$. In particular, $\mathcal{S}$ is 1-Lipschitz with respect to $l2$-nom : $\forall x, z \in \mathbb{R}^d, \|\mathcal{S}(x) - \mathcal{S}(z)\|_2 \leq \|x - z\|_2$. This property provides exact robustness certificates for OCSDF in the form of a certified AUROC that can be computed at the same cost as AUROC. This regularity translates into solid empirical robustness as compared to other OCC baselines. In other words, OCSDF alleviates the **lack of negative data** and the **robustness issue**. We go further and highlight interesting research perspectives regarding OCSDF. Indeed, we show that learning the SDF with a 1-Lipschitz network enables a generative procedure that allows visualizing points at the boundary of $\mathcal{X}$. Moreover, It implicitly parametrizes the shape of $\mathcal{X}$, which connects One-Class Classification with implicit surface parametrization, intensively used in computer graphics for shape reconstruction.

Our contributions are as follows. **(1)** We introduce a new OCC framework based on the Signed Distance Function to the boundary of the data distribution. We theoretically demonstrate that the SDF can be learned with a 1-Lipschitz neural net using the Hinge Kantorovich-Rubinstein (HKR) loss and Negative Sampling; **(2)** We evaluate the performances of OCSDF on several benchmarks and show its benefits for theoretical and empirical robustness; and **(3)** we demonstrate how OCSDF extends the applications of One Class Classification from traditional OOD detection to generative visualization and implicit surface parametrization for shape reconstruction from point clouds.

Contents

5.1 Related Work on One Class classification

One Class Classification (OCC) OCC is an instance of binary classification where all the points of the dataset at hand belong to the same (positive) class. The challenge of this task is to construct a decision boundary without using points from the other (negative) class. OCC amounts to finding a domain containing the support of the data distribution. That is why OCC is mainly used in Out Of Distribution (OOD), anomaly or novelty detection, with positive samples considered In Distribution (ID) and negative ones as OOD, anomalies or novelties. This task dates back to Sager (1979); Hartigan (1987) and was popularized for anomaly detection with One-class Support Vector Machines (OC-SVM) (Schölkopf et al., 1999). Since then, the field of OCC has flourished with many well-established algorithms such as Local Outlier Factors (Breunig et al., 2000), Isolation Forests (Liu et al., 2008) and their variants (see Han et al. (2022) for a thorough benchmark). More recently, since Deep-SVDD Ruff et al. (2018) - followed by several works such as Bergman and Hoshen (2019); Golan and El-Yaniv (2018); Goyal et al. (2020); Zenati et al. (2018); Sabokrou et al. (2018) - Deep Learning has emerged as a relevant alternative to perform OCC due to its capacities to handle large dimensional data. However, methods of this field suffer from their lack of **robustness and certifications**, which makes them vulnerable to adversarial attacks. In addition, they always struggle to cope with the **lack of OOD data**. In this paper, we tackle these problems with an OCC algorithm based on approximating the SDF using **1-Lipschitz neural nets**. In addition, the SDF being intensively used in Computer Graphics, our algorithm establishes a new link between OCC and **implicit surface** parametrization.

SDF for neural implicit surfaces Historically, signed distance functions have been used in computer graphics to parametrize a surface as the level set of some function (Novello et al., 2022). Given an incomplete or unstructured representation of a geometrical object (like a 3D point cloud or a triangle soup), recent methods aim at representing a smooth shape either as vectors in the latent space of a generative model (Achlioptas et al., 2018; Ben-Hamu et al., 2018; Groueix et al., 2018; Chou et al., 2022) or directly as parameters of a neural net (Park et al., 2019; Atzmon and Lipman, 2020). The first method allows for easy shape interpolation, while the latter proved to be a more robust approach (Davies et al., 2021). Those neural implicit surfaces alleviate both the problems related to memory requirements of voxel-based representations and the combinatorial nature of meshes, making them ideally suited for rendering using ray marching (Hart, 1995) and constructive solid geometry. In those contexts, the constraint $\|\nabla_x f(x)\| \leq 1$ is necessary to guarantee the validity of the geometrical query while having $\|\nabla_x f(x)\|$ as close as possible to 1 allows for greedier queries and faster computation times. In practice, training an SDF requires a dataset (p, d) of points $p \in \mathbb{R}^3$ with their corresponding signed distance d to the desired surface. Computing those distances requires the existence and availability of a ground truth, which is not always the case. Moreover, training tends to be unstable in general, and special care is needed for most computer graphics applications (Sharp and Jacobson, 2022). Our method can instead be trained to approximate a surface without prior knowledge of the distances and is provably robust.

Tackling the lack of OOD data The previously mentioned OCC and OOD algorithms, as well as many others (Hendrycks and Gimpel, 2018; Hsu et al., 2020) are designed to avoid the need for OOD data. However, some works aim at falling back to classical binary classification by artificially generating negative samples. The idea of Negative Sampling is not recent and appeared in Forrest et al. (1994) for detecting computer viruses or to emulate the distinction made by antibodies between pathogens and body cells (Gonzalez et al., 2002). It has been introduced in anomaly detection by Ayara et al. (2002) and studied by several works summarized in Jinyin and Dongyong (2011), but

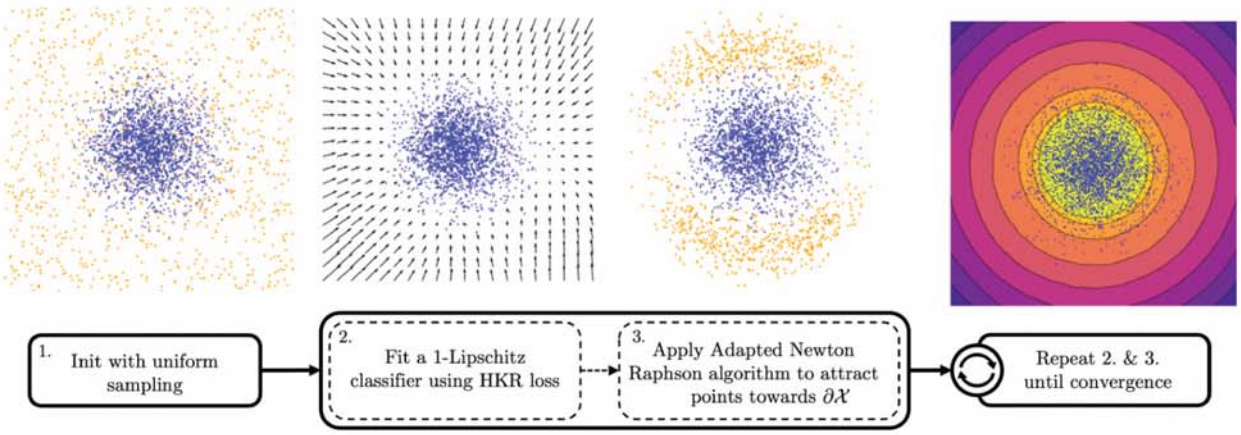


Figure 5.1: **Summary of One Class Signed Distance Function (OCSDF)**. We start with an uniform negative sampling, then we fit a 1-Lipschitz classifier f_θ using the Hinge Kantorovich-Rubinstein loss. We apply the Adapted Newton Raphson algorithm 6 to attract the points towards the boundary of the domain $\partial\mathcal{X}$ thanks to the smoothness of f_θ, which in addition allows providing robustness certificates.

has lost popularity due to its practical inefficiency (e.g. compared to One-Class Support Vector Machines (OCSVM) (Stibor et al., 2005)). Recently, some works revived the idea of using OOD data, either by artificial negative sampling (Lee et al., 2018b; Sipple, 2020; Goyal et al., 2020; Pourreza et al., 2021), or by using OOD data from other sources, a procedure called outlier exposure (Fort et al., 2021; Hendrycks et al., 2019). However, outlier exposure suffers from bias since OOD data does not come from the same data space. Therefore, we follow the first idea and sample negative data points close to the domain $\mathcal{X}$, thanks to the orthogonal neural nets-based estimation of the SDF.

5.2 Semi-supervised learning of Signed Distance Functions

This method aims to learn the Signed Distance Function (SDF) by reformulating the one-class classification of $\mathbb{P}_X$ as a binary classification of $\mathbb{P}_X$ against a carefully chosen distribution $Q(\mathbb{P}_X)$. We show that this formulation yields desirable properties, especially when the chosen classifier is a 1-Lipschitz neural net trained with the Hinge Kantorovich-Rubinstein (HKR) loss.

5.2.1 SDF learning formulated as binary classification

The idea is to learn one class classifier by reformulating one class learning of $\mathbb{P}_X$ as a binary classification of $\mathbb{P}_X$ against a carefully chosen adversarial distribution $Q(\mathbb{P}_X)$, as defined below.

Definition 17 ($\overset{B,\epsilon}{\sim}$ Complementary Distribution)**.** *Let $\mathbb{P}_X$ a distribution with compact support $\mathcal{X} \subset B$, with $B \subset \mathbb{R}^d$ a bounded measurable set. Q is said to be (B, ϵ) disjoint from $\mathbb{P}_X$ if (i) its support supp $Q \subset B$ is compact (ii) $d(\text{supp } Q, \mathcal{X}) \geq 2\epsilon$ (iii) for all measurable sets $M \subset B$ such that $d(M, \mathcal{X}) \geq 2\epsilon$ we have $Q(M) > 0$. It defines a symmetric but irreflexive binary relation denoted $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$.*

Exemple 5.1. Domain B.

For image space with pixel intensity in $[0,1]$, we take $B = [0,1]^{W\times H\times C}$. For tabular data, a hypercube of side length ten times the standard deviation of the data along the axis. A data point falling outside B is trivially considered anomalous due to aberrant values.

This simple idea had already occurred repeatedly in the related literature (Sabokrou et al., 2018). Note that $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ benefits from generalization guarantees as proved in chapter 4: the optimal classifier on the train set and on the test set are the same in the limit of big samples.

Binary classification between $\mathbb{P}_X$ and any $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$ allows the construction of the optimal signed distance function, using the Kantorovich-Rubinstein (HKR) Hinge loss (Serrurier et al., 2021), thanks to the following theorem.

Theorem 2. SDF Learning with HKR loss. *Let $\mathcal{L}^{hkr}_{m,\lambda}(yf(x)) = \lambda \max(0, m - yf(x)) - yf(x)$ be the Hinge Kantorovich Rubinstein loss, with margin $m = \epsilon$, regularization $\lambda > 0$, prediction $f(x)$ and label $y \in \{-1, 1\}$. Let Q be a probability distribution on B. Assume that λ is high enough. Let $\mathcal{E}^{hkr}(f)$ be the population risk:*

$$\mathcal{E}^{hkr}(f, \mathbb{P}_X, Q) := \mathbb{E}_{x\sim\mathbb{P}_X}[\mathcal{L}^{hkr}_{m,\lambda}(f(x))] + \mathbb{E}_{z\sim Q}[\mathcal{L}^{hkr}_{m,\lambda}(-f(z))]. \tag{5.2}$$

Let f^* be the minimizer of population risk, whose existence is guaranteed with Arzelà-Ascoli theorem:

$$f^* \in \underset{f\in Lip_1(\mathbb{R}^n,\mathbb{R})}{\arg\inf} \mathcal{E}^{hkr}(f, \mathbb{P}_X, Q), \tag{5.3}$$

*where $Lip_1(\mathbb{R}^n, \mathbb{R})$ is the set of Lipschitz functions $\mathbb{R}^d \to \mathbb{R}$ of constant 1. Assume that $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$. **Then**, f^* approximates the signed distance function over B:*

$$\begin{aligned} \forall x \in \mathcal{X}, \quad & \mathcal{S}(x) = f^*(x) - m, \\ \forall z \in supp\ Q, \quad & \mathcal{S}(z) = f^*(z) - m. \end{aligned} \tag{5.4}$$

Moreover, for all $x \in supp\ Q \cup \mathcal{X}$:

$$sign(f(x)) = sign(\mathcal{S}(x)).$$

Proof. The results follow from the properties of $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss given in Proposition 2 of Serrurier et al. (2021). If $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$, then by definition, the two datasets are 2ϵ separated. Consequently the hinge part of the loss is null: $\max(0, m - yf(x))$ for all pairs $(x, +1)$ and $(z, -1)$ with $x \sim \mathbb{P}_X$ and $z \sim Q$. We deduce that:

$$\forall x \in \mathcal{X}, \qquad f(x) \geq m, \tag{5.5}$$

$$\forall z \in \text{supp } Q, \qquad f(z) \leq -m. \tag{5.6}$$

In the following we use the notations:

$$F_z := \bigcup_{x\in\mathcal{X}} \underset{z_0\in\text{supp } Q}{\arg\min} \|x - z_0\|_2,$$

and

$$F_x := \bigcup_{z\in(\text{supp } Q)} \underset{x_0\in\mathcal{X}}{\arg\min} \|x_0 - z\|_2.$$

Since $m = \epsilon$, we must have $f(x) = m$ for all $x \in F_x$, and $f(z) = -m$ for all $z \in F_z$, whereas $\mathcal{S}(x) = 0$ and $\mathcal{S}(z) = -2m$. Thanks to the 1-Lipschitz property for every $x \in \mathcal{X}$ we have $f(x) \leq f(\partial x) + \|x - \partial x\|$ where $\partial x = \arg\min_{\bar{x} \in \partial\mathcal{X}} \|x - \bar{x}\|$ is the projection of x onto the boundary $\partial\mathcal{X}$. Similarly $f(z) \geq f(\partial z) - \|z - \partial z\|$. The $-yf(x)$ term in the $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss (Wasserstein regularization), incentives to maximize the amplitude $|f(x)|$ so the inequalities are tight. Notice that $\mathcal{S}(x) = \mathcal{S}(\partial x) + \|x - \partial x\|$ and $\mathcal{S}(z) = \mathcal{S}(\partial z) - \|z - \partial z\|$. This allows concluding:

$$\forall x \in \mathcal{X}, \qquad \mathcal{S}(x)) = f^*(x) - m, \tag{5.7}$$

$$\forall z \in \text{supp } Q, \qquad \mathcal{S}(z) = f^*(z) - m. \tag{5.8}$$

Remark: the Proposition 2 found in Serrurier et al. (2021) mentions the condition $\lambda \geq 0$ but this seems to be a mistake. Indeed, when $\lambda = 0$ the hinge term does not play a role. By continuity, it seems unlikely that the result would be true for arbitrary small λ. On the other hand, for λ high enough and separable classes, it seems reasonable that the hinge part of the loss drops to zero. Therefore, the proof of this proposition should be fixed if its result is required in the future. □

Note that if $m = \epsilon \ll 1$, then we have $f^*(x) \approx \mathcal{S}(x)$. In this work, we parametrize f as a 1-Lipschitz neural network, because they fulfil $f \in \text{Lip}_1(\mathbb{R}^n, \mathbb{R})$ by definition.

Theorem 2 tells us that if we characterize the complementary distribution Q, we can approximate the SDF with a 1-Lipschitz neural classifier trained with HKR loss. We now need to find the complementary distribution Q.

5.2.2 Complementary distribution generation

We propose to seek Q through an alternating optimization process: at every iteration t, a proposal distribution Q_t is used to train a 1-Lipschitz neural net classifier f_t against $\mathbb{P}_X$ by minimizing empirical HKR loss. Then, the proposal distribution is updated in Q_{t+1} based on the loss induced by f_t, and the procedure is repeated. We suggest starting from the uniform distribution: $Q_0 = \mathcal{U}(B)$.

Remark 5.1. Concentration phenomenon

Observe that in high dimension, due to the *curse of dimensionality*, a sample $z \sim Q_0$ is unlikely to satisfy $z \in \mathcal{X}$. Indeed the data lies on a low dimensional manifold $\mathcal{X}$ for which the Lebesgue measure is negligible compared to B. Hence, in the limit of small sample size $n \ll \infty$, a sample $Z_n \sim Q_0^{\otimes n}$ fulfills $Z_n \overset{B,\epsilon}{\sim} \mathbb{P}_X$. This phenomenon is called the Concentration Phenomenon and has already been leveraged in anomaly detection in Sipple (2020). However, the *curse* works both ways and yields a high variance in samples Z_n. Consequently, the variance of the associated minimizers f_0 of equation 5.3 will also exhibit a high variance, which may impede the generalization and convergence speed. Instead, the distribution Q_t must be chosen to produce higher density in the neighborhood of the boundary $\partial\mathcal{X}$.

The true boundary is unknown, but the level set $\mathbb{L}_t = f_t^{-1}(\{-\epsilon\})$ of the classifier can be used as a proxy to improve the initial proposal Q_0. We start from $z_0 \sim Q_0$ and then look for a displacement $\delta \in \mathbb{R}^d$ such that $z + \delta \in \mathbb{L}_t$. To this end, we take inspiration from the multidimensional Newton-Raphson method and consider a linearization of f_t:

$$f_t(z_0 + \delta) \approx f_t(z_0) + \langle \nabla_x f_t(z_0), \delta \rangle. \tag{5.9}$$

Since 1-Lipschitz neural nets with GroupSort activation function are piecewise *affines* Tanielian and Biau (2021), the linearization is locally exact, hence the following property.

Property 5. *Let f_t be a 1-Lipschitz neural net with GroupSort activation function. Almost everywhere on $z_0 \in \mathbb{R}^d$, there exists $\delta_0 > 0$ such that for every $\|\delta\| \leq \delta_0$, we have:*

$$f_t(z_0 + \delta) = f_t(z_0) + \langle \nabla_x f_t(z_0), \delta \rangle. \tag{5.10}$$

Since $f_t(z_0 + \delta) \in \mathbb{L}_t$ translates into $f_t(z_0 + \delta) = -\epsilon$,

$$\delta = -\frac{f_t(z_0) + \epsilon}{\|\nabla_x f_t(z_0)\|^2} \nabla_x f_t(z_0). \tag{5.11}$$

Properties of $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ ensure that the optimal displacement follows the direction of the gradient $\nabla_x f_t(z_0)$, which coincides with the direction of an optimal transportation plan (Serrurier et al., 2021). The term $\|\nabla_x f_t(z_0)\|$ enjoys an interpretation as a Local Lipschitz Constant (see Jordan and Dimakis (2020)) of f_t around z_0, which we know fulfills $\|\nabla_x f_t(z_0)\| \leq 1$ when parametrized with an 1-Lipschitz neural net. When f_t is trained to perfection, the expression for δ simplifies to $\delta = -f_t(z_0)\nabla_x f_t(z_0)$ thanks to Property 6.

Property 6 (Minimizers of $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ are Gradient Norm Preserving (Serrurier et al., 2021))**.** *Let f_t^* be the solution of Equation 5.3. Then for almost every $z \in B$ we have $\|\nabla_x f_t^*(z)\| = 1$.*

In practice, the exact minimizer f_t^* is not always retrieved, but equation 5.11 still applies to imperfectly fitted classifiers. The final sample $z' \sim Q_t$ is obtained by generating a sequence of T small steps to smooth the generation. The procedure is summarized in algorithm 6. In practice, T can be chosen very low (below 16) without significantly hurting the quality of generated samples. This has also been observed in the context of Langevin dynamics used for the training of Energy Based Models (Teh et al., 2003) like noticed in Nijkamp et al. (2019, 2020).

Remark 5.2. behavior when Q_t and $\mathbb{P}_X$ overlap

Consider a sample $z \in \mathbb{L}_t$. If z is a *false positive* (i.e $f_t(z) > 0$ and $z \notin \mathcal{X}$), training f_{t+1} on the pair $(z, -1)$ will incentive f_{t+1} to fulfill $f_{t+1}(z) < 0$, which will reduce the volume of false positive associated to f_{t+1}. If z is a *true negative* (i.e. $f_t(z) < 0$ and $z \notin \mathcal{X}$) it already exhibits the wanted properties. The case of *false negative* (i.e $f_t(z) < 0$ and $z \in \mathcal{X}$) is more tricky: the density of $\mathbb{P}_X$ around z will play an important role to ensure that $f_{t+1}(z) > 0$. This motivates the introduction of assumption 1.

We assume that samples from the target $\mathbb{P}_X$ are *significantly* more frequent than the ones obtained from pure randomness. It is a very reasonable assumption (especially for natural images, for example), and most distributions from real use cases fall under this setting.

Assumption 1 ($\mathbb{P}_X$ samples are more frequent than pure randomness (informal))**.** *For any measurable set $M \subset \mathcal{X}$ we have $\mathbb{P}_X(M) \gg \mathcal{U}(M)$, where $\mathcal{U}$ is the uniform distribution over B.*

To ensure that property (iii) of definition 17 is fulfilled, we also introduce stochasticity in algorithm 6 by picking a random "learning rate" $\eta \sim \mathcal{U}([0, 1])$ for each negative example in the batch. This ensures they distribute evenly on the path toward the boundary.

The final procedure is depicted in algorithm 5. The procedure also benefits from Property 7, which ensures that the distribution Q_{t+1} obtained from Q_t across several iterative applications of Algorithm 6 still fulfills $Q^{t+1} \overset{B,\epsilon}{\sim} \mathbb{P}_X$. It guarantees that once the complementary distribution has been found, the algorithm will continue to produce a sequence of complementary distributions and a sequence of classifiers f_t that approximates $\mathcal{S}$.

Property 7 (Complementary distributions are fix points)**.** *Let Q^t be such that $Q^t \overset{B,\epsilon}{\sim} \mathbb{P}_X$. Assume that Q^{t+1} is obtained with algorithm 5. Then we have $Q^{t+1} \overset{B,\epsilon}{\sim} \mathbb{P}_X$.*

Proof. The proof also follows from the properties of $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss given in Proposition 2 of Serrurier et al. (2021) Since $Q_t \overset{B,\epsilon}{\sim} \mathbb{P}_X$ all examples $z \sim Q_t$ generated fulfill (by definition) $d(z, \mathcal{X}) \geq 2\epsilon \geq 2m$. Indeed the 1-Lipschitz constraint (in property 6) guarantees that no example z_t can "overshoot" the boundary. Hence for the associated minimizer f_{t+1} of $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss, the hinge part of the loss is null. This guarantees that $f_{t+1}(z) \leq -m$ for $z \sim Q$. We see that by applying algorithm 6 the property is preserved: for all $z \sim Q_{t+1}$ we must have $f_{t+1}(z) \leq -m = -\epsilon$. Finally notice that because $z_0 \sim \mathcal{U}(B)$ and $\eta \sim \mathcal{U}([0,1])$ the support supp Q covers the whole space B (except the points that are less than 2ϵ apart from $\mathcal{X}$). Hence we have $Q_{t+1} \overset{B,\epsilon}{\sim} \mathbb{P}_X$ as expected. □

Algorithm 5 Alternating Minimization for Signed Distance Function learning

Input: 1-Lipschitz neural network architecture $f_\circ$
Input: initial weights θ_0, learning rate α

repeat
  $f_t \leftarrow f_{\theta_t}$
  $\tilde{\theta} \leftarrow \theta_t$
  repeat
    Generate batch $z \sim Q_t$ of negative samples with algorithm 6
    Sample batch $x \sim \mathbb{P}_X$ of positive samples
    Compute loss on batch $\mathcal{L}(\tilde{\theta}) := \mathcal{E}^{\text{hkr}}(f_{\tilde{\theta}}, x, z)$
    Learning step $\tilde{\theta} \leftarrow \tilde{\theta} + \alpha \nabla_\theta \mathcal{L}(\tilde{\theta})$
  until convergence of $\tilde{\theta}$ to θ_{t+1}.
until convergence of f_t to limit f^*.

Algorithm 6 Adapted Newton–Raphson for Complementary Distribution Generation

Input: 1-Lipschitz neural net f_t
Parameter: number of steps T
Output: sample $z' \sim Q_t(f)$

sample learning rate $\eta \sim \mathcal{U}([0,1])$
$z_0 \sim \mathcal{U}(B)$ ▷ Initial approximation.
for each step $t = 1$ to T **do**
  $z_{t+1} \leftarrow z_t - \frac{\eta}{T} \frac{\nabla_x f(z^t)}{\|\nabla_x f(z^t)\|_2^2} (f(z_t) + \epsilon)$ ▷ Refining.
  $z_{t+1} \leftarrow \Pi_B(z_{t+1})$ ▷ Stay in feasible set.
end for
return z_T

5.2.3 Lazy variant with amortized optimization

Algorithm 5 solves a MaxMin problem with alternating maximization-minimization. The inner minimization step (minimization of the loss over 1-Lipschitz function space) can be expensive. Instead, partial minimization can be performed by doing only a predefined number of gradient steps. This yields the lazy approach of algorithm 7. This provides a considerable speed up over

the initial implementation. Moreover, this approach is frequently found in literature, for example, with GAN (Goodfellow et al., 2014) or WGAN (Arjovsky et al., 2017). This can also be seen as an instance of *amortized optimization* (Amos et al., 2023). However, we lose some of the mild guarantees, such as the one of Proposition 7 or even Theorem 2. Crucially, it can introduce unwanted oscillations in the training phase that can impede performance and speed. Hence this trick should be used sparingly.

Algorithm 7 Signed Distance Function learning: lazy approach.

Input: 1-Lipschitz neural net architecture $f_\circ$, initial weights θ_0, learning rate α, number of parameter update per time step K

repeat
  $\tilde{\theta} \leftarrow \theta_t$
  Generate batch $z \sim Q_t$ of negative samples with algorithm 6
  Sample batch $x \sim \mathbb{P}_X$ of positive samples
  for K updates **do**
    Compute loss on batch $\mathcal{L}(\theta) := \mathcal{E}^{\text{hkr}}(f_{\tilde{\theta}}, x, z)$
    Learning step $\tilde{\theta} \leftarrow \tilde{\theta} + \alpha \nabla_\theta \mathcal{L}(\tilde{\theta})$
  end for
  $\theta_{t+1} \leftarrow \tilde{\theta}$
until convergence of f_t.

The procedure of algorithm 7 bears numerous similarities with the adversarial training of Madry et al. (2018). In our case the adversarial examples are obtained by starting from noise $\mathcal{U}(B)$ and relabeled as *negative* examples. In their case, the adversarial examples are obtained by starting from $\mathbb{P}_X$ itself and relabeled as *positive* examples.

5.2.4 Sampling the potential

Remark 5.3. density of the complementary distribution

In high dimension $d \gg 1$, when $\|\nabla_x f_t(z)\| = 1$ and $\text{Vol}(B) \gg \text{Vol}(\mathcal{X})$ the samples obtained with algorithm 6 are approximately uniformly distributed on the levels sets of f_t. Intuitively, the level sets of the SDF for some connected set $\mathcal{X}$ can be seen as a homeomorphism of a sphere. Therefore, the volume of the level set $(\cup_{t_0 \leq t \leq t_0+w} \mathbb{L}_t)$ of (small) width w growths as $\mathcal{O}(wt_0^d)$, which implies that the density of Q increases exponentially fast (with factor d) with respect to the value of $-|f_t(\cdot)|$. This mitigates the adverse effects of the curse of dimensionality. This behavior is quantitatively different from the case where the density is uniform.

This scheme of "generating samples by following gradient in input space" reminds diffusion models (Ho et al., 2020), feature visualization tools (Olah et al., 2017), or recent advances in VAE (Kuzina et al., 2022) or Iterative VAEs (Boutin et al., 2020). However, no elaborated scheme is required for the training of f_t: 1-Lipschitz networks exhibit smooth and interpretable gradients (Serrurier et al., 2023) which allows sampling from $\mathcal{X}$ "for free" as illustrated in figure 5.2.

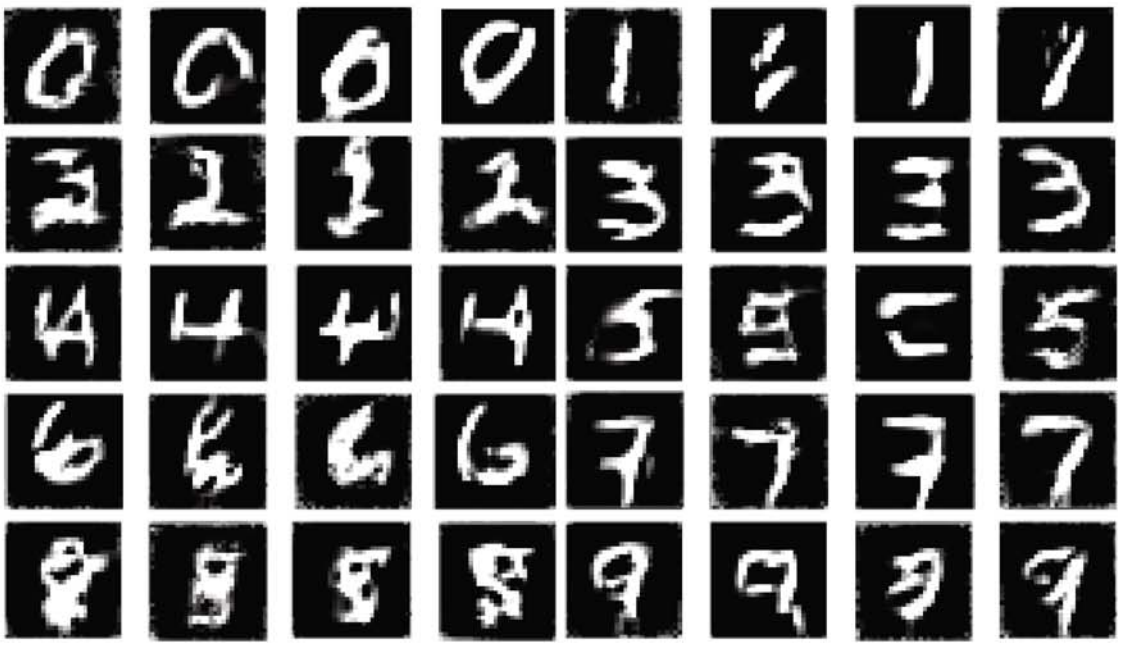


Figure 5.2: **Syntetic examples** generated from algorithm 6 with $T = 64$ and $\eta = 1$.

Remark 5.4. Langevin dynamics

A more precise characterization of Q_t built with algorithm 6 can be sketched below. Our approach bears some similarities with the spirit of Metropolis-adjusted Langevin algorithm (Grenander and Miller, 1994). In this method, the samples of $p(x)$ are generated by taking the stationary distribution $x_{t\to\infty}$ of a continuous Markov chain obtained from the stochastic gradient step iterates

$$x_{t+1} \leftarrow x_t + \zeta \nabla_x \log p(x) + \sqrt{2\zeta} Z \tag{5.12}$$

for some distribution $p(x)$ and $Z \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. By choosing the level set $\epsilon = 0$, and $p(x) \propto \mathbb{1}\{f(x) \leq 0\} \exp\left(-\eta f^2(x)\right)$ the score function $\zeta \nabla_x \log p(x)$ is transformed into $\nabla_x f(x)|f(x)|$ with $\zeta = \frac{\eta}{T}$. Therefore, we see that the density decreases exponentially faster with the squared distance to the boundary $\partial\mathcal{X}$ when there are enough steps $T \gg 1$. In particular when $\mathcal{X} = \{0\}$ we recover $p(x)$ as the pdf of a standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{1})$. Although the similarity is not exact (e.g., the diffusion term $\sqrt{2\eta}Z$ is lacking, T is low, $\eta \sim \mathcal{U}([0, 1])$ is a r. v.), it provides interesting complementary insights on the algorithm.

5.2.5 Alternating minimization for SDF learning

Each classifier f_t does not need to be trained from scratch. Instead, the same architecture is kept throughout training, and the algorithm produces a sequence of parameters θ_t such that $f_t = f_{\theta_t}$. Each set of parameters θ_t is used as initialization for the next one θ_{t+1}. Moreover, we only perform a low fixed number of parameter updates for each t in a GAN fashion. The final procedure of OCSDF is shown in Figure 5.1.

5.3 Applications to implicit surface parametrization

Our approach to learning the SDF contrasts with the computer graphics literature, where SDF is used to obtain the distance of a point to a surface (here defined as $\partial\mathcal{X}$). Indeed, SDFs are usually learned in a supervised fashion, requiring the ground truth of l_2 distance. This is classically achieved using Nearest-Neighbor algorithms, which can be cumbersome, especially when the number of points is high. Efficient data structures , e.g., using K-dtrees (Maneewongvatana and Mount, 1999) or Octrees (Meagher, 1980), can mitigate this effect but do not scale well to high dimensions. Instead, OCSDF learns the SDF solely based on points contained in the support. While neural network training is not cheap by any mean, the network approach is advantageous at inference time. Indeed, with a dataset of size n, a single forward in the network costs $\mathcal{O}(1)$ (furthermore orthogonalization of matrices can be done once for all), while naive implementations of kNN costs $\mathcal{O}(n^2)$, or at least $\mathcal{O}(n \log n)$ for more effiient implementations with trees.

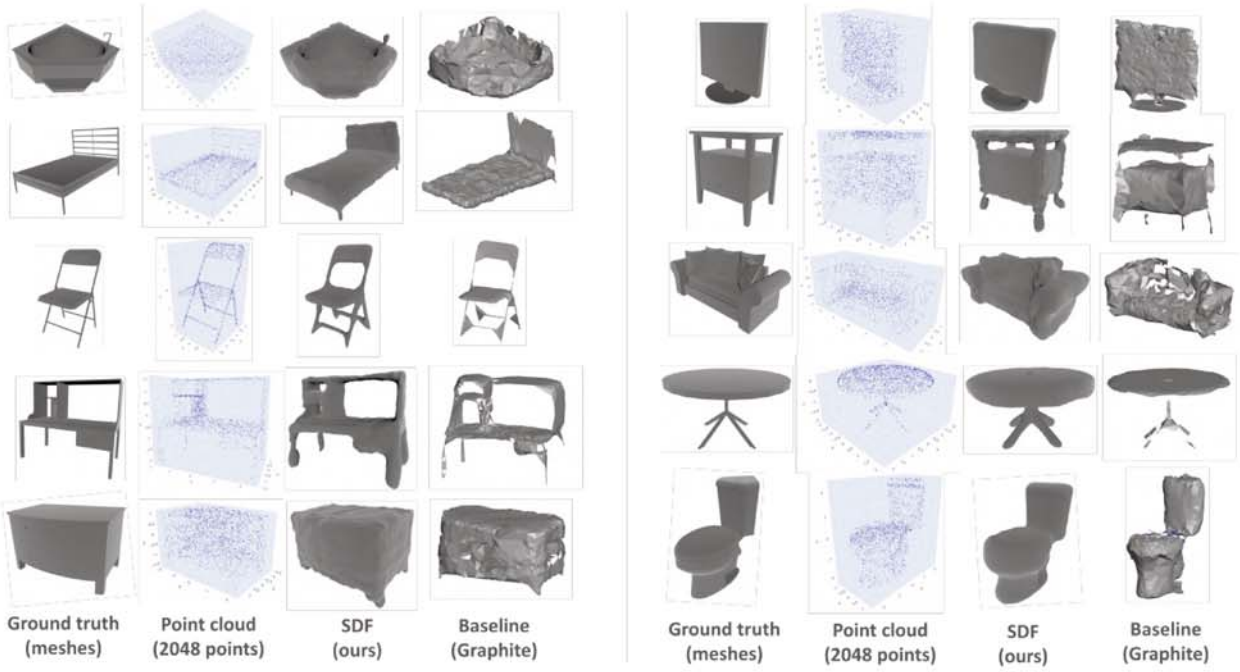


Figure 5.3: **Visualization of the Implicit Surface**. **First column**: ground truth meshes. **Second column**: sparse point clouds of size 2048 sampled with *Trimesh* library. **Third column**: our method. **Fourth column**: a baseline, the SSSR algorithm Boltcheva and Lévy (2017) that attempts to reproduce the meshes solely from a point cloud, and distributed in the package *Graphite*. The SDF exhibits better extrapolation properties and provides smooth surfaces.

Exemple 5.2. implicit surface parametrization on ModelNet10 dataset

We use models from Princeton's ModelNet10 dataset Wu et al. (2015). We sample $n = 2048$ points within each mesh to obtain a 3D point cloud. We fit the SDF on the point cloud with the same hyperparameters as the tabular experiment. We use Lewiner marching algorithm (Lewiner et al., 2003) from scikit-image (Van der Walt et al., 2014), on a $200 \times 200 \times 200$ voxelization of the input. We plot the mesh reconstructed with Trimesh (Dawson-Haggerty et al., 2019).

The results are highlighted in figure 5.3. We chose the first percentile of $\mathbb{E}_{x\sim\mathbb{P}_X}[f(x)]$ as the level set of the iso-surface we plot. We compare our results visually against a baseline from Boltcheva and Lévy (2017) implemented in Graphite Levy (2022) that rebuilds the mesh solely from the point cloud (without extrapolation). We highlight that $n = 2048$ is considered low resolution; hence many details are expected to be lost. Nonetheless, our method recovers the global aspect of the shape more faithfully, smoothly, and consistently than the other baseline.

In figure 5.4, we provide additional examples of the use of SDF to reconstruct shapes from point clouds. We also compare OCSDF with Deep SDF (Park et al., 2019), a standard baseline for neural implicit surface parametrization, to highlight the practical advantages of our method.

Remark 5.5. Groupsort and compositionality of signed distance functions.

If f and g are the signed distance function of the shapes S_f and S_g respectively, then $\min f, g$ is the SDF of $S_f \cup S_g$, and $\max$ (see Bálint et al. (2023) for example). Observe that our 1-Lipschitz network relies on GroupSort activation function, composed of min and max gates. Therefore, each non-linearity in the network is a union or intersection of other shapes. The linear layers are orthogonal, so together with the bias they define a "rigid transformation", i.e. a combination of rotations, symmetries, and translations.

Methods	OCSDF (ours)	DeepSDF
Target	Generate $Q \overset{B,\epsilon}{\sim} P$	Compute $\mathcal{S}(x)$ from P
Cost	Backward pass	Nearest Neighbor search
Loss	HKR $\mathcal{L}^{\text{hkr}}_{m,\lambda}$	$\|f(x) - \mathcal{S}(x)\|_2^2$
Guarantees	f is 1-Lipschitz	None

Table 5.1: **Comparison of our approach against DeepSDF** (Park et al., 2019).

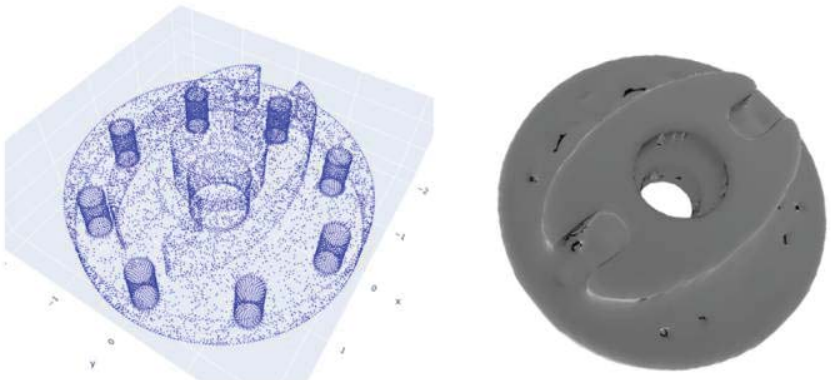


Figure 5.4: **SDF of** 2048 **points sampled from the mesh of a 3D CAD model.** This task is particularly challenging due to the number of holes in the surface, that the SDF tends to "fill".

5.4 Robust One Class learning

We conclude the experimental part with the initial motivations of the work: robust One Class classification and anomaly detection.

5.4.1 Certificates against adversarial attacks

The most prominent advantage of 1-Lipschitz neural nets is their ability to produce certificates against adversarial attacks (Szegedy et al., 2014). Indeed, by definition we have $f(x+\delta) \in [f(x) - \|\delta\|, f(x) + \|\delta\|]$ for every example $x \in \mathcal{X}$ and every adversarial attack $\delta \in \mathbb{R}^d$. This allows bounding the changes in AUROC score of the classifier for every possible radius $\epsilon > 0$ of adversarial attacks.

Proposition 12 (certifiable AUROC)**.** *Let F_0 be the cumulative distribution function associated with the negative classifier's prediction (when $f(x) \leq 0$), and p_1 the probability density function of the positive classifier's prediction (when $f(x) > 0$). Then, for any attack of radius $\epsilon > 0$, the AUROC of the attacked classifier f_ϵ can be bounded by*

$$AUROC_\epsilon(f) = \int_{-\infty}^{\infty} F_0(t) p_1(t - 2\epsilon) dt. \tag{5.13}$$

Proof. Let p_1 (resp. p_{-1}) be the probability density function (PDF) associated with the classifier's positive (resp. negative) predictions. More precisely, p_1 (resp. p_{-1}) is the PDF of $f_\sharp \mathbb{P}_X$ (resp. $f_\sharp Q$) for some adversarial distribution Q, where $f_\sharp \cdot$ denotes the pushforward measure operator (Bogachev and Ruas, 2007) defined by the classifier. The operator $f_\sharp$ formalizes the shift between $\mathbb{P}_X$ (resp. Q), the ground truth distributions, and p_{-1} (resp. p_1), the imperfectly distribution fitted by f. Let F_{-1} and F_1 be the associated cumulative distribution functions. For a given classification decision threshold τ, we can define the True Positive Rate (TPR) $F_{-1}(\tau)$, the True Negative Rate (TNR) $1 - F_1(\tau)$, and the False Positive Rate (FPR) $F_1(\tau)$. The ROC curve is then the plot of $F_{-1}(\tau)$ against $F_1(\tau)$. Hence, setting $v = F_1(t)$, we can define the AUROC as:

$$\text{AUROC}(f) = \int_0^1 F_{-1}(F_1^{-1}(v)) dv. \tag{5.14}$$

And with the change of variable $dv = p_1(t)dt$ we get

$$\text{AUROC}(f) = \int_{-\infty}^{\infty} F_{-1}(t) p_1(t) dt. \tag{5.15}$$

We consider a scenario with symmetric attacks: the attack decreases (resp. increases) the normality score of One Class (resp. Out Of Distribution samples) for decision threshold $\tau \in \mathbb{R}$. When the 1-Lipschitz classifier f is under attacks of radius at most $\epsilon > 0$ we note f_ϵ the perturbed classifier:

$$f_\epsilon(x) = \min_{\delta \leq \epsilon} (2\mathbb{1}\{f(x) \geq \tau\} - 1) f(x + \delta). \tag{5.16}$$

Note that $f_\epsilon(x) \leq f(x) + \epsilon$ when $f(x) < \tau$ and $f_\epsilon(x) \geq f(x) - \epsilon$ when $f(x) \geq \tau$ thanks to the 1-Lipschitz property. This effectively translates the p.d.f of f_ϵ by $|\epsilon|$ atmost.

We obtain a lower bound for the AUROC (i.e a certificate):

$$\begin{aligned} \text{AUROC}(f_\epsilon) &\geq \int_{-\infty}^{\infty} F_{-1}(t + \epsilon) p_1(t - \epsilon) dt \\ &= \int_{-\infty}^{\infty} F_{-1}(t) p_1(t - 2\epsilon) dt. \end{aligned} \tag{5.17}$$

□

Takeaways

The certified AUROC score can be computed analytically without performing the attacks empirically, solely from score predictions $p_1(t - 2\epsilon)$. More importantly, the certificates hold against *any* adversarial attack whose l_2-norm is bounded by ϵ, regardless of the algorithm used to perform such attacks. We emphasize that producing certificates is more challenging than traditional defence mechanisms (e.g, adversarial training, see Bai et al. (2021) and references therein) since they do not target defence against a specific attack method. Note that MILP solvers and branch-and-bound approaches (Tjeng et al., 2019; Wang et al., 2021) can be used to improve the tightness of certificates, but at a higher cost.

5.4.2 One Class learning on images

We evaluate the performances of OCSDF for OCC, where only samples of the normal class are supposed to be available. To emulate this setting, we train a classifier on each of the classes of MNIST and Cifar10, and evaluate it on an independent test set in a *one-versus-all* fashion. Note that the *out-of-distribution* examples are not seen during training, but more importantly, the *in-distribution* examples from the test set are not seen either. Hence the task evaluates both the generalization capacity (new example from the *in-distribution*) and the discriminative capacity (against *out-of-distribution*). This setting is challenging because of the curse of dimensionality.

We compare our method against DeepSVDD (Ruff et al., 2018), OCSVM (Schölkopf et al., 1999), and Isolation Forests (Liu et al., 2008). The mean AUROC score is reported in table 5.2 and averaged over 20 runs. It is computed between the $1,000$ test examples of the target class and the remaining $9,000$ examples from other classes of the test set (both unseen during training). OCSDF is competitive against other baselines. In addition, it comes with several advantages described in the following.

Certifiable and empirical robustness

None of the concurrent methods can provide certificates against l_2 attacks: in the work of Goyal et al. (2020) the attacks are performed empirically (no certificates) with l_∞ radii. In table 5.2, we report our certifiable AUROC with various radii $\epsilon \in \{0, 8/25, 16/255, 36/255, 72/255\}$. In figure 5.5 we report the empirical AUROC against l_2-PGD attacks with three random restarts, using stepsize $\zeta = 0.025\epsilon$ like in the default policy of Foolbox (Rauber et al., 2020). These results illustrate our method's benefits: not only does it come with robustness certificates that are verified empirically, but the empirical robustness is also way better than DeepSVDD, especially for Cifar10. Note that for 1-Lipschitz network trained with $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss, all the attacks tend to find the same adversaries (Serrurier et al., 2021) - hence PGD is also representative of the typical score that would have been obtained with other attack methods.

Visualization of the support

OCSDF can be seen as a parametric version of kNN, which enables this approach in high dimensions. As a result, the decision boundary learned by the classifier can be materialized by generating adversarial examples with algorithm 6. The forward computation graph is a classifier based on optimal transport, and the backward computation graph is an image generator. Indeed, the back-propagation through a convolution is a transposed convolution, a popular layer in the generator of

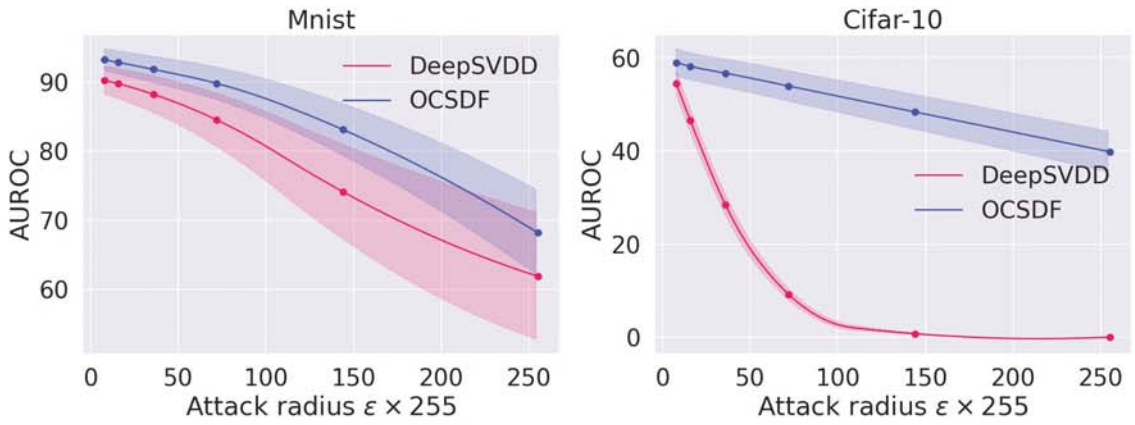


Figure 5.5: **Empirical Mean AUROC on all classes against adversarial attacks** of various radii in One Class setting, using default parameters of FoolBox Rauber et al. (2020).

GANs. Overall, the algorithm behaves like a WGAN (Arjovsky et al., 2017) with a single network fulfilling both roles. This unexpected feature opens a path to the explainability of the One Class classifier: the support learned can be visualized without complex feature visualization tools. In particular, it helps identify failure modes.

5.5 Anomaly detection and nearest neighbors

One Class (OC) learning and Anomaly Detection (AD) are, *technically*, two different tasks. The former belongs to the supervised learning paradigm (the goal is to learn the support of a distribution from samples), while the latter belongs to the unsupervised learning paradigm. In anomaly detection the "".

5.5.1 Toy examples

We use two-dimensional toy examples from the Scikit-Learn library (Pedregosa et al., 2011). Results are shown in figure 5.6. The contour of the decision function are plotted in resolution 300×300 pixels. The level sets of the classifier are compared against those of One Class SVM (Schölkopf et al., 2001b) and Isolation Forest (Liu et al., 2008). We also train a conventional network with Binary Cross Entropy against complementary distribution Q_t, and we show it struggles to learn a meaningful decision boundary. Moreover, its Local Lipschitz Constant (Jordan and Dimakis, 2020) increases uncontrollably, as shown in table 5.3, which makes it prone to adversarial attacks. Finally, there is no natural interpretation of the prediction of the conventional network in terms of distance: the magnitude $|f(\cdot)|$ of the predictions quickly grows above 10^3, whereas for 1-Lipschitz neural nets, it is approximately equal to the signed distance function $\mathcal{S}$.

5.5.2 Anomaly Detection on Tabular datasets

We tested our algorithm on some of the most prominent anomaly detection benchmarks of ODDS library (Rayana, 2016). In this unsupervised setting (like ADBench Han et al. (2022)) all the examples (normal examples and anomalies) are seen during training, but their true label is unknown. To apply our method, the only hyperparameter needed is the margin m that we select in the range $[0.01, 0.05, 0.2, 1.]$. For each value, the results are averaged over 20 independent runs train/test splits.

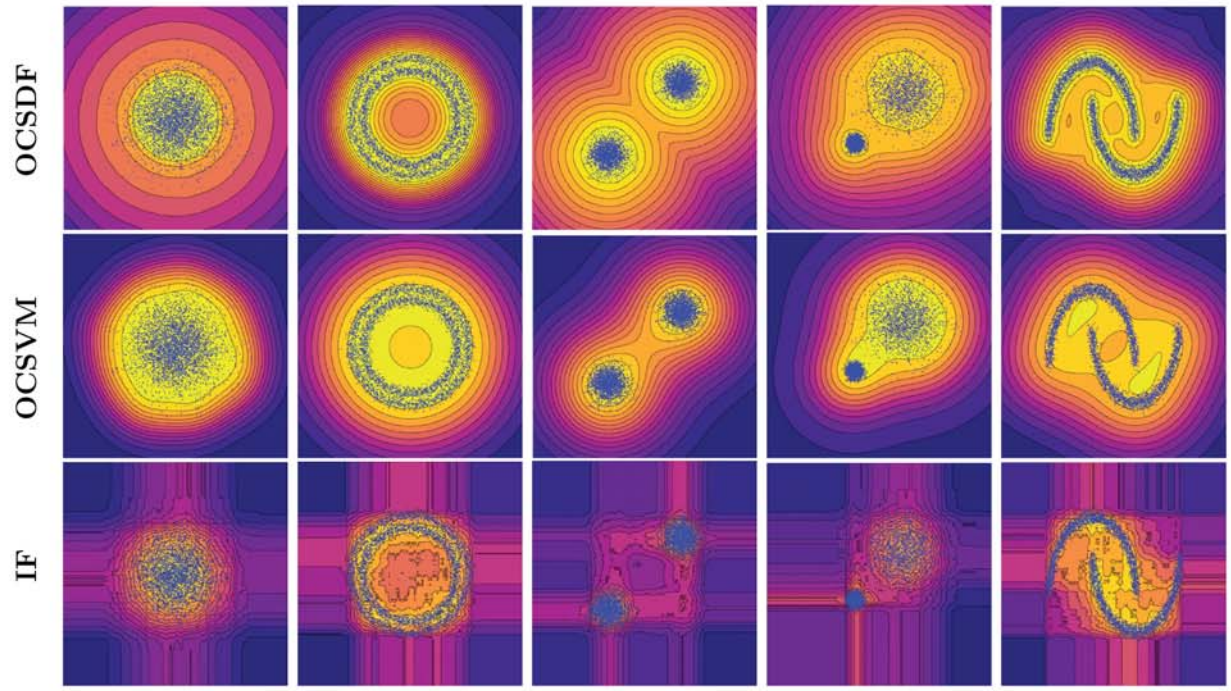
MNIST	OCSDF	OCSDF	OCSDF	OCSDF	OCSDF	OC SVM	Deep SVDD	IF
Certificates	$\epsilon = 0$	$\epsilon = 8/255$	$\epsilon = 16/255$	$\epsilon = 36/255$	$\epsilon = 72/255$	$\epsilon = 0$	$\epsilon = 0$	$\epsilon = 0$
mAUROC	$\mathbf{95.5 \pm 0.4}$	93.2 ± 2.1	89.9 ± 3.5	78.4 ± 6.4	57.5 ± 7.5	91.3 ± 0.0	$\mathbf{94.8 \pm 0.9}$	92.3 ± 0.5
digit 0	$\mathbf{99.7 \pm 0.1}$	99.6 ± 0.2	99.5 ± 0.2	99.0 ± 0.6	96.2 ± 3.0	98.6 ± 0.0	98.0 ± 0.7	98.0 ± 0.3
digit 1	$\mathbf{99.8 \pm 0.0}$	99.7 ± 0.0	99.6 ± 0.1	99.2 ± 0.3	96.2 ± 1.6	99.5 ± 0.0	$\mathbf{99.7 \pm 0.1}$	97.3 ± 0.4
digit 2	$\mathbf{90.6 \pm 2.0}$	85.3 ± 1.9	78.2 ± 2.3	53.1 ± 5.2	14.1 ± 4.6	82.5 ± 0.1	$\mathbf{91.7 \pm 0.1}$	88.6 ± 0.5
digit 3	$\mathbf{93.4 \pm 1.2}$	90.0 ± 1.7	85.0 ± 2.3	66.2 ± 4.6	26.9 ± 5.0	88.1 ± 0.0	91.9 ± 1.5	89.9 ± 0.4
digit 4	$\mathbf{96.5 \pm 0.9}$	95.3 ± 1.2	93.9 ± 1.7	89.4 ± 3.6	76.2 ± 9.8	94.9 ± 0.0	$\mathbf{94.9 \pm 0.8}$	92.7 ± 0.6
digit 5	$\mathbf{93.9 \pm 2.2}$	89.0 ± 3.2	81.6 ± 4.7	54.0 ± 8.7	15.6 ± 6.9	77.1 ± 0.0	88.5 ± 0.9	85.5 ± 0.8
digit 6	$\mathbf{98.7 \pm 0.6}$	98.1 ± 0.7	97.2 ± 0.9	93.1 ± 2.6	74.9 ± 10.4	96.5 ± 0.0	$\mathbf{98.3 \pm 0.5}$	95.6 ± 0.3
digit 7	$\mathbf{97.1 \pm 0.6}$	96.5 ± 0.5	95.6 ± 0.6	92.2 ± 0.8	81.2 ± 1.7	93.7 ± 0.0	94.6 ± 0.9	92.0 ± 0.4
digit 8	89.4 ± 2.6	83.3 ± 5.1	74.7 ± 9.0	50.3 ± 15.9	24.4 ± 14.0	88.9 ± 0.0	$\mathbf{93.9 \pm 1.6}$	89.9 ± 0.4
digit 9	$\mathbf{96.4 \pm 0.3}$	95.3 ± 0.9	93.8 ± 1.3	87.8 ± 3.1	68.9 ± 7.6	93.1 ± 0.0	$\mathbf{96.5 \pm 0.3}$	93.5 ± 0.3

CIFAR10	OCSDF	OCSDF	OCSDF	OCSDF	OCSDF	OC SVM	Deep SVDD	IF
Certificates	$\epsilon = 0$	$\epsilon = 8/255$	$\epsilon = 16/255$	$\epsilon = 36/255$	$\epsilon = 72/255$	$\epsilon = 0$	$\epsilon = 0$	$\epsilon = 0$
mAUROC	57.4 ± 2.1	53.1 ± 2.1	48.8 ± 2.1	38.4 ± 1.9	22.5 ± 1.4	64.8 ± 8.0	64.8 ± 6.8	55.4 ± 8.0
Airplane	$\mathbf{68.2 \pm 4.5}$	64.3 ± 3.9	60.1 ± 3.2	49.4 ± 1.1	31.2 ± 3.6	61.6 ± 0.9	61.7 ± 4.1	60.1 ± 0.7
Automobile	57.3 ± 1.7	52.5 ± 3.0	47.6 ± 4.2	36.1 ± 6.8	19.8 ± 7.8	$\mathbf{63.8 \pm 0.6}$	$\mathbf{65.9 \pm 2.1}$	50.8 ± 0.6
Bird	$\mathbf{51.8 \pm 2.7}$	47.5 ± 1.8	43.2 ± 1.6	33.4 ± 3.4	19.5 ± 5.7	$\mathbf{50.0 \pm 0.5}$	$\mathbf{50.8 \pm 0.8}$	$\mathbf{49.2 \pm 0.4}$
Cat	$\mathbf{58.8 \pm 1.2}$	54.6 ± 0.8	50.3 ± 0.8	40.0 ± 1.5	24.4 ± 2.3	55.9 ± 1.3	$\mathbf{59.1 \pm 1.4}$	55.1 ± 0.4
Deer	49.4 ± 2.4	45.3 ± 2.1	41.4 ± 1.9	32.2 ± 1.5	18.8 ± 1.4	$\mathbf{66.0 \pm 0.7}$	60.9 ± 1.1	49.8 ± 0.4
Dog	56.3 ± 0.6	51.9 ± 1.0	47.5 ± 1.6	36.7 ± 2.9	20.6 ± 4.0	62.4 ± 0.8	$\mathbf{65.7 \pm 2.5}$	58.4 ± 0.5
Frog	52.6 ± 1.8	48.7 ± 1.7	44.9 ± 1.6	35.8 ± 1.4	22.4 ± 1.1	$\mathbf{74.7 \pm 0.3}$	67.7 ± 2.6	42.9 ± 0.6
Horse	49.5 ± 0.9	45.5 ± 1.0	41.5 ± 1.2	32.5 ± 1.5	18.8 ± 1.6	62.6 ± 0.6	$\mathbf{67.3 \pm 0.9}$	55.1 ± 0.7
Ship	68.6 ± 1.8	64.6 ± 1.4	60.4 ± 1.3	49.3 ± 2.4	29.8 ± 4.9	$\mathbf{74.9 \pm 0.4}$	$\mathbf{75.9 \pm 1.2}$	$\mathbf{74.2 \pm 0.6}$
Truck	61.3 ± 3.4	56.5 ± 2.1	51.5 ± 1.1	39.0 ± 3.9	20.0 ± 7.0	$\mathbf{75.9 \pm 0.3}$	73.1 ± 1.2	58.9 ± 0.7

Table 5.2: **AUROC score on the test set of MNIST and CIFAR10** in a *one versus all* fashion, averaged on 10 runs. We also report the AUROC of DeepSVDD Ruff et al. (2018) for completeness, along with the other AUROC scores of Isolation Forest (IF) and One-Class SVM (OC-SVM) reported in Ruff et al. (2018). When the differences between some methods are not statistically significant, we highlight both. When the confidence intervals overlap, we highlight both. We also show the **certifiable** AUROC against l-2 attacks of norms $\epsilon \in \{8/255, 16/255, 36/255\}$. Concurrent methods cannot provide certificates for $\epsilon > 0$.

Empirical Local Lipschitz Constant	One Cloud	Two Clouds	Two Blobs	Blob Cloud	Two Moons
	26.66	122.84	1421.41	53.90	258.73

Table 5.3: **Lower bound on the Local Lipschitz Constant (LLC)** of conventional network after 10,000 training steps for each toy example. It is the maximum of $\|\nabla_{x_i} f(x_i)\|_2$ over the train set.



(a) One cloud. (b) Two circles. (c) Two blobs. (d) Blob & cloud. (e) Two moons.

Figure 5.6: **Toy examples from Scikit-learn**. **Top row**: our method with Lipschitz (LIP) 1-Lipschitz network and $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ (HKR) loss. **Second row**: One Class SVM. **Third row**: Isolation Forest.

Following ADBench guidelines and best practices from the AD community, we only compute the AUROC, since this metric is symmetric under label flip. We report the best average in table 5.4 along baselines from ADBench Han et al. (2022). As observed by Han et al. (2022), none of the algorithms clearly dominates the others, because what is considered an anomaly (or not) depends on the context. Among 14 other methods tested, our algorithm ranks $7.1 \pm 3.6/15$, while the best (Isolation Forests) ranks $4.5 \pm 3.2/15$. The experiment shows that our algorithm is competitive with respect to other broadly used baselines. Nonetheless, it brings several additional advantages. First, our algorithm can be seen as a parametric version of kNN for the euclidean distance, which leverages deep learning to avoid the costly construction of structures like a KDTree Maneewongvatana and Mount (1999) and the quadratic cost of nearest neighbor search, thereby enabling its application in high dimensions. Second, it provides robustness certificates.

Dataset	d	#no.+an.	perc.	OCSDF (Ours)	Deep SVDD	OC SVM	IF	PCA	kNN	SOTA
breastw	9	444+239	35%	(#10) 82.6 ± 5.9	65.7	80.3	98.3	95.1	97.0	99.7 (COPOD)
cardio	21	1,655+176	9.6%	(#2) 95.0 ± 0.1	59.0	93.9	93.2	95.5	76.6	95.5 (PCA)
glass	9	205+9	4.2%	(#7) 73.9 ± 4.1	47.5	35.4	77.1	66.3	82.3	82.9 (CBLOF)
http (KDDCup99)	3	565,287+2,211	0.4%	(#11) 67.5 ± 37	69.0	99.6	99.96	99.7	03.4	99.96 (IF)
Ionosphere	33	225+126	36%	(#7) 80.2 ± 0.1	50.9	75.9	84.5	79.2	88.3	90.7 (CBLOF)
Lymphography	18	142+6	4.1%	(#8) 96.1 ± 4.9	32.3	99.5	99.8	99.8	55.9	99.8 (CBLOF)
mammography	6	10,923+260	2.32%	(#6) 86.0 ± 2.5	57.0	84.9	86.4	88.7	84.5	90.7 (ECOD)
musk	166	2,965+97	3.2%	(#8) $92.6 \pm 20.$	43.4	80.6	99.99	100.0	69.9	100.0 (PCA)
Optdigits	64	5,066+150	3%	(#12) 51.0 ± 0.9	38.9	54.0	70.9	51.7	41.7	87.5 (CBLOF)
Pima	8	500+268	35%	(#12) 60.7 ± 1.0	51.0	66.9	72.9	70.8	73.4	73.4 (kNN)
satimage-2	36	5,732+71	1.2%	(#3) 97.9 ± 0.4	53.1	97.3	99.2	97.6	92.6	99.8 (CBLOF)
Shuttle	9	45,586+3,511	7%	(#4) 99.1 ± 0.3	52.1	97.4	99.6	98.6	69.6	99.6 (IF)
smtp (KDDCup99)	3	95,126+30	0.03%	(#4) 87.1 ± 3.5	78.2	80.7	89.7	88.4	89.6	89.7 (IF)
speech	400	3,625+61	1.65%	(#15) 46.0 ± 0.2	53.4	50.2	50.7	50.8	51.0	56.0 (COF)
thyroid	6	3,679+93	2.5%	(#5) 95.9 ± 0.0	49.6	87.9	98.3	96.3	95.9	98.3 (IF)
vertebral	6	210+30	12.5%	(#4) 48.6 ± 2.6	36.7	38.0	36.7	37.0	33.8	53.2 (DAGMM)
vowels	12	1,406+50	3.4%	(#2) 94.7 ± 0.7	52.5	61.6	73.9	65.3	97.3	97.3 (kNN)
WBC	30	357+21	5.6%	(#10) 93.6 ± 0.1	55.5	99.0	99.0	98.2	90.6	99.5 (CBLOF)
Wine	13	119+10	7.7%	(#5) 81.5 ± 0.9	59.5	73.1	80.4	84.4	45.0	91.4 (HBOS)
Average Rank among all tasks				7.1 ± 3.6	11.2	7.5	4.5	5.7	7.8	4.5 ± 3.2 (IF)

Table 5.4: **AUROC score for tabular data**, averaged over 20 runs. The dimension of the dataset is denoted by d. In the **Anomaly Detection protocol (AD)** we use all the data (normal class and anomalies) for training, in an unsupervised fashion. The "#no.+an." column indicates part of normal (no.) and anomalous (an.) data used during training for each protocol. SOTA denominates the best score ever reported on the dataset, obtained by crawling relevant literature, or ADBench (Han et al., 2022) results (table D4 page 37). We report the rank as (#rank) among 14 other methods.

5.6 Lipschitz Energy-Based-Models

In this section, we explore further the link between SDF learning and Energy Based Models (Teh et al., 2003; Du and Mordatch, 2019). These preliminary results are unpublished. The goal of energy-based models is to learn the probability density functions of the data $\mathbb{P}_X$. In this setting, the network f_θ parametrizes an energy function $E_\theta(x) := \exp -f_\theta(x)$, i.e. an *un-normalized* probability distribution. Indeed, the normalization factor

$$Z = \int_B \exp -f_\theta(x) \mathrm{d}x \tag{5.18}$$

is too expensive to compute at each step since it involves an integral over the whole domain B.

Fortunately, it is possible to sample from $p_\theta \propto E_\theta$ using Langevin Dynamics Grenander and Miller (1994), as explained in the previous section. The goal is to minimize the negative log-likelihood of the data $\mathcal{L}(\theta) := \mathbb{E}_{x\sim\mathbb{P}_X}[-\log(p_\theta(x))]$, whose gradient simplifies to:

$$\nabla_\theta \mathcal{L}^{\mathrm{kr}}(\theta) = \mathbb{E}_{x\sim p_\theta}[\nabla_\theta E_\theta(x)] - \mathbb{E}_{x\sim\mathbb{P}_X}[\nabla_\theta E_\theta(x)]. \tag{5.19}$$

Once again, we recognize the Kantorovich-Rubinstein loss between $\mathbb{P}_X$ and p_θ. This time, there are no Lipschitz constraints, and like in section 4.2 the problem is not stationary: p_θ is always moving. Intuitively, gradient steps behave as if they are trying the minimize the distance between p_θ and $\mathbb{P}_X$. Note that the Langevin dynamics attempts to sample from p_θ by following gradient $\nabla_x f_\theta(x)$, whereas Algorithm 6. We highlight below the similarities and differences between the two algorithms.

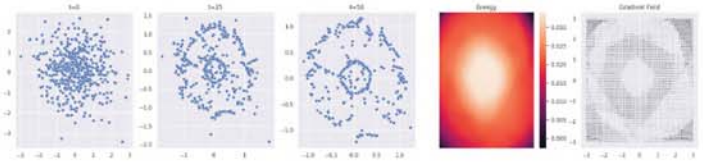
	OC-SDF (ours)	**EBM** (Du and Mordatch, 2019)
Goal	Learn SDF $\mathcal{S}$	Learn density of $\mathbb{P}_X$
Approximator	LipNet1 network f_θ	Energy $E_\theta(x)$
Loss	Minimize $\mathcal{L}^{\mathrm{hkr}}_{m,\lambda}$	Maximize likelihood
Sampling	Complementary $Q \overset{B,\epsilon}{\sim} \mathbb{P}_X$	$p_\theta \approx \mathbb{P}_X$ from potential E_θ
Sampler	Newton-Raphson	Stochastic Langevin

Remark 5.6. Compositionality rules of SDF and EBM.

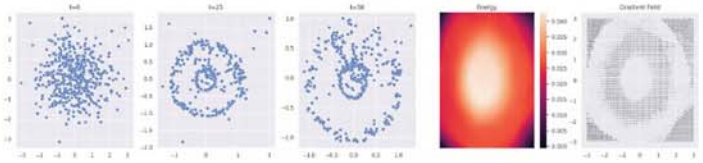
Interestingly, both signed distance functions and energy-based models support compositionality: complex scenes and probability distributions can be modelized as a certain composition of simpler structure. This permits efficient training of individual components, and then they can be effortlessly merged. In the context of rendering, this allows one to describe a scene as a set of objects, each with its own SDF. In the context of data science, this allows one to describe a dataset as a union of sub-datasets, for which adding, removing or changing data sources is transparent. Once again, the links between the two approaches deserve to be drawn.

	Signed Distance Function	**Energy Based Models**
Target	**Implicit surface** S_f, S_g	**Distributions** p_f, p_g
Approximators	Functions f, g	Energies E_f, E_g
Union	$\min(f, g)$	$-\mathrm{LogSumExp}(-E_f, -E_g)$[a]
Intersection	$\max(f, g)$	$E_f + E_g$
Set Difference	$\max(-f, g)$	$-E_f + E_g$

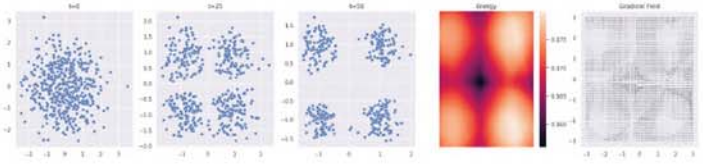
In this table we used the common convention of negative sign inside the shape for the SDF, unlike the beginning of the chapter. The data on EBM come from (Du et al., 2020, 2023).



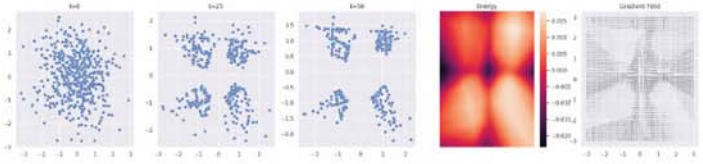
(a) **Two Circles**, conventional network.



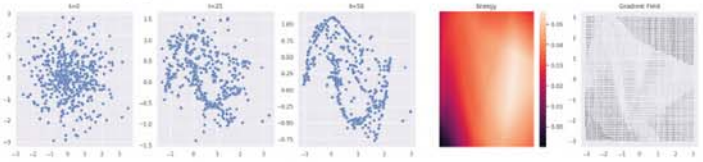
(b) **Two Circles**, Lipschitz network.



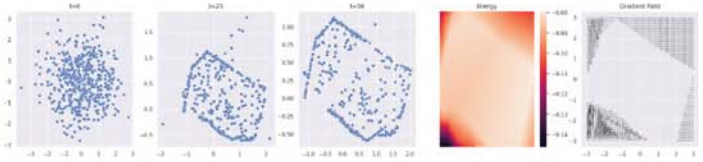
(c) **Four Blobs**, conventional network.



(d) **Four Blobs**, Lipschitz network.



(e) **Two Moons**, conventional network.



(f) **Two Moons**, Lipschitz network.

Figure 5.7: **EBM training with a conventional network (left) or 1-Lipschitz network (right)**. The first three panels are points sampled from Langevin dynamics at steps $t \in [0, 25, 50]$. The fourth panels are the level sets of the log-energy function f_θ. The fifth panel shows the gradient field $\nabla_x f_\theta$.

Observe that even if f_θ is K-Lipschitz, the function $\exp -f_\theta$ is not K-Lipschitz, and the associated probability p_θ can vanish exponentially quickly outside of the support of $\mathbb{P}_X$. Therefore, Lipschitz constraints on f_θ are less severe than it seems.

[a] It is actually a weighted mixture. Controlling the weight requires to know the normaization constants Z_f, Z_g in advance, which is untractable.

Exemple 5.3. Smooth energy-based models with Lipschitz functions.

In Figure 5.7 we train:

- a conventional AllNet network.
- a 1-Lipschitz LipNet1 network.

to learn the energy of the Two Moons dataset. Experiments are run in Pytorch. The hope is that, like for SDF, the Enerby-Based model may benefit from the smoothness of the score f_θ to stabilize sampling from Langevin dynamics, and avoid catastrophic divergence. Evaluation of EBMs, like may generative models, is not easy, and just "taking a look" at what have been generated is often one of the best (subjective) metrics. Based on this subjective appreciation, these preliminary results are encouraging. A more objective measure would be to compute the Wasserstein distance between the samples from p_θ and $\mathbb{P}_X$.

The link between implicit surface parametrization and EBM is also studied in the very recent work of Yamauchi et al. (2023).

5.7 Perspectives

Despite its performance and appealing properties, the method suffers from some important limitations we highlight below and that can serve as a basis for future work.

5.7.1 Eikonal networks approximation power

The performance of the algorithm strongly depends on its capacity to properly learn the true minimizer f^* of $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss. Per Property 6 such minimizer must fulfill $\|\nabla_x f^*(x)\|_2 = 1$ everywhere on the support of $\mathbb{P}_X$ and Q_t. Hence the performance of the algorithm (and the associated theoretical guarantees) depends on the capacity of the GNP network to fulfil this property. In the tabular case, it is easy to do using orthogonal matrices for affine layers and GroupSort/FullSort (Anil et al., 2019)) activation functions. However, in the image case, designing "orthogonal convolution" is still an active research area. Several solutions have been proposed, but they come with various drawbacks in terms of simplicity of implementation, computational cost, or tightness of the constraint. Hence the average gradient norm on image datasets struggles to exceed 0.3 in practice. Another limitation stems from low-rank adjoint operators (e.g the last layer of the network): during backpropagation they do not preserve gradient norm along all directions. The Newton-Raphson trick that uses steps of size $\frac{\nabla_x f(x)}{\|\nabla_x f(x)\|_2^2}$ mitigates partially the issue. This suggests that the algorithm (in its current form) could benefit from further progress in Gradient Norm Preserving architectures.

5.7.2 Limitations of the Euclidean norm in image space

The algorithm provides metric guarantees in the construction of the Signed Distance Function (SDF) to the boundary. The $l2$-norm is not a crucial component of the construction: the proof of 2 and Proposition 2 of Serrurier et al. (2021) can be applied to any norm. However, in every case, the Lipschitz constraint $|f(x) - f(y)| \leq \|x - y\|_L$ on the network architecture must coincide with the norm $\|\cdot\|_L$ used to build the signed distance function. Currently, only networks that are Lipschitz with respect to l^∞ and l^2 norms benefit from universal approximation properties (Anil et al., 2019). Those norms are often meaningful for tabular data, but not for images. Hence, metric guarantees are less useful in pixel space. The method still benefits from certificates against adversarial attacks, which is highly desirable for critical systems but lacks semantic interpretation otherwise.

5.7.3 Tuning of the margin

The algorithm is not quite agnostic to the data: the margin $m > 0$ used in $\mathcal{L}^{\text{hkr}}_{m,\lambda}$ loss is an important parameter that serves as prior on the typical distance that separates the One Class support from the anomalies. This hyper-parameter can be guessed from the final application at hand (very much like the "scale" parameter of radial basis function kernels), manually with grid search algorithms or more extensive procedures. Theorem 2 suggests that a small margin m works best. However, the VC dimension associated with the corresponding set of classifiers increases polynomially with $\frac{1}{m}$ (see Proposition 6 of Béthune et al. (2022)). Hence, the algorithm benefits from faster convergence and more stability during training when m is big. Fortunately, this tradeoff present in most deep learning-based algorithms is solely controlled by this one-dimensional parameter in our case. Any heuristic estimation from data or with a one-dimensional line search is feasible, even with a limited computational budget.

5.7.4 Conclusion

This work showed the promising approach of applying Lipschitz-constrained neural networks to the field of computer graphics. Including these Lipschitz constraints, and more specifically the Eikonal condition $\|\nabla_x f\| = 1$ to signed distance function is currently a hot-topic: we can mention the very recent works of Ma et al. (2023) or Yang et al. (2023).

Chapter 6

Lipschitzness with respect to parameters and application to differential privacy

In this chapter, we show how to derive Lipschitzness w.r.t parameters from Lipschitz bounds on input. We illustrate these properties in the context of training deep neural networks with differential privacy guarantees. This chapter is mostly adapted from

Louis Béthune, Thomas Masséna, Thibaut Boissin, Yannick Prudent, Corentin Friedrich, Franck Mamalet, Aurelien Bellet, Mathieu Serrurier, David Vigouroux, **DP-SGD Without Clipping: the Lipschitz neural network way**, preprint, *2023*.

Differential privacy allows to develop methods for training models that preserve the privacy of individual data points in the training set. The field of differential privacy seeks to enable deep learning on sensitive data, while ensuring that models do not inadvertently memorize or reveal specific details about individual samples in their weights. This involves incorporating privacy-preserving mechanisms into the design of deep learning architectures and training algorithms, whose most popular example is Differentially Private Stochastic Gradient Descent (Abadi et al., 2016).

Contents

6.1 Differential privacy

Differential privacy. Informally, differential privacy (DP) is a *definition* that bounds how much the change of a single sample in a dataset affects the range of a stochastic function (here, the training algorithm). This definition is largely accepted as a strong guarantee against privacy leakages under various scenarii, including data aggregation or post-processing (Dwork et al., 2006).

In this chapter we focus on supervised learning tasks: we assume that the dataset $\mathcal{D}$ is a finite collection of input/label pairs $\mathcal{D} = \{(x_1, y_1), \ldots ...(x_N, y_N)\}$. The definition of DP relies on the notion of neighboring datasets, i.e datasets that vary by at most one example. We highlight below the central tools related to the field, inspired from Dwork et al. (2014).

Definition 18 ((ϵ, δ)-Approximate Differential Privacy)**.** *Two datasets $\mathcal{D}$ and $\mathcal{D}'$ are said to be neighboring for the "add/remove-one" relation if they differ by exactly one sample: $|\mathcal{D}' \ominus \mathcal{D}| = 1$ where $\ominus$ denotes the symmetric difference between sets. Let ϵ and δ be two non-negative scalars. An algorithm $\mathcal{A}$ is (ϵ, δ)-DP if for any two neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$, and for any $S \subseteq range(\mathcal{A})$:*

$$\mathbb{P}[\mathcal{A}(\mathcal{D}) \in S] \leq e^{\epsilon} \times \mathbb{P}[\mathcal{A}(\mathcal{D}') \in S] + \delta. \tag{6.1}$$

A popular rule of thumb suggests using $\epsilon \leq 10$ and $\delta < \frac{1}{N}$ with N the number of records (Ponomareva et al., 2023) for mild guarantees. In practice, most classic algorithmic procedures (called *queries* in this context) do not readily fulfill the definition for useful values of (ϵ, δ): in particular, randomization is mandatory. A general recipe to make a query differentially private is to compute its *sensitivity* Δ, and to perturb its output by adding a Gaussian noise of predefined variance $\zeta^2 = \Delta^2\sigma^2$, where the (ϵ, δ) guarantees depend on σ, yielding what is called a *Gaussian mechanism* (Dwork et al., 2006).

Definition 19 (l_2-sensitivity)**.** *Let $\mathcal{M}$ be a query mapping from the space of the datasets to $\mathbb{R}^p$. Let $\mathcal{N}$ be the set of all possible pairs of neighboring datasets $\mathcal{D}, \mathcal{D}'$. The l_2 sensitivity of $\mathcal{M}$ is defined by:*

$$\Delta(\mathcal{M}) = \sup_{\mathcal{D}, \mathcal{D}' \in \mathcal{N}} \|\mathcal{M}(D) - \mathcal{M}(D')\|_2. \tag{6.2}$$

This randomization comes at the expense of "utility", i.e the usefulness of the output for downstream tasks (Alvim et al., 2012). The goal is then to strike a balance between privacy and utility, ensuring that the released information remains useful and informative for the intended purpose while minimizing the risk of privacy breaches. The privacy/utility trade-off yields a Pareto front, materialized by plotting ϵ against a measurement of utility, such as validation accuracy for a classification task.

Differentially Private SGD. The SGD algorithm consists of a sequence of queries that (i) take the dataset in input, sample a minibatch from it, and return the gradient of the loss evaluated on the minibatch, before (ii) performing a descent step following the gradient direction. In "add-remove" neighboring relations, if the gradients are bounded by $K > 0$, the sensitivity of the gradients averaged on a minibatch of size b is $\Delta = K/b$. DP-SGD (Abadi et al., 2016) makes each of these queries private by resorting to the Gaussian mechanism. Crucially, the algorithm requires a bound on gradient norms $\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2 \leq C$. This upper bound on gradient norms is generally unknown in advance, which leads practitioners to clip it to $C > 0$, in order to bound the sensitivity manually. Unfortunately, this creates a number of issues: **1.** Hyper-parameter search on the broad-range clipping value C is required to train models with good privacy/utility trade-offs (Papernot and Steinke, 2022), **2.** The computation of per-sample gradients is expensive: DP-SGD is usually slower and consumes more memory than vanilla SGD, in particular for the large batch sizes often used in private training (Lee

```
model = DP_Sequential([ # step 1: use DP_Sequential to build a model
        # step 2: add Lipschitz layers of known sensitivity
        DP_BoundedInput(input_shape=(28, 28, 1), upper_bound=20.),
        DP_SpectralConv2D(filters=16, kernel_size=3, use_bias=False),
        DP_GroupSort(2),
        DP_Flatten(),
        DP_SpectralDense(1)],
    dp_parameters=dp_parameters,
    dataset_metadata=dataset_metadata,
) # step 4: compile the model, and choose any first order optimizer
model.compile(loss=DP_TauBCE(tau=20.), optimizer=Adam(1e-3))
model.fit( # step 5: train the model and measure the DP guarantees
    train_dataset, validation_data=val_dataset,
    epochs=num_epochs, callbacks=[DP_Accountant()]
)
```

Figure 6.1: **An example of usage of our framework**, illustrating how to create a small Lipschitz VGG and how to train it under (ϵ, δ)-DP guarantees while reporting (ϵ, δ) values.

and Kifer, 2021), **3.** Clipping the per-sample gradients biases their average (Chen et al., 2020). This is problematic as the average direction is mainly driven by misclassified examples.

An unexplored approach: Lipschitz constrained networks. To avoid these issues, we propose to train neural networks for which the parameter-wise gradients are provably and analytically bounded during the whole training procedure, in order to get rid of the clipping process. This allows for efficient training of models without the need for tedious hyper-parameter optimization. The main reason why this approach has not been experimented much in the past is that upper bounding the gradient of neural networks is often intractable. However, by leveraging the literature on Lipschitz constrained networks introduced by Anil et al. (2019), we show that these networks have computable bounds for their gradient's norm. This yields tight bounds on the sensitivity of SGD steps, making their transformation into Gaussian mechanisms inexpensive - hence the name **Clipless DP-SGD**.

The literature has predominantly focused on investigating the control of Lipschitzness with respect to the inputs (i.e bounding $\nabla_x f$), primarily motivated by concerns of robustness (Szegedy et al., 2014; Li et al., 2019a; Fazlyab et al., 2019), or improved generalization (Bartlett et al., 2017; Béthune et al., 2022). However, in this work, we will demonstrate that it is also possible to control Lipschitzness with respect to parameters (i.e. bounding $\nabla_\theta f$), which is essential for ensuring privacy. Our first contribution will point out the tight link that exists between those two quantities. The closest work to ours is Shavit and Gjura (2019), where a Lipschitz network is used as a general function approximator of bounded sensitivity $\|\nabla_x f(\theta, x)\|_2$, but to this day the sensitivity of the gradient query $x \mapsto \nabla_\theta f(\theta, x)$ itself remains largely unexplored. The idea of using automatic differentiation to compute sensitivity bounds has also been discussed in Ziller et al. (2021) and Usynin et al. (2021).

Contributions. While the properties of Lipschitz-constrained networks regarding their inputs are well explored, the properties with respect to their parameters remain non-trivial. This work provides a first step to fill this gap: our analysis shows that under appropriate architectural constraints, a l-Lipschitz network has a tractable, finite Lipschitz constant with respect to its parameters, which allows for easy estimation of the sensitivity of the gradient computation queries. Our contributions are the following:

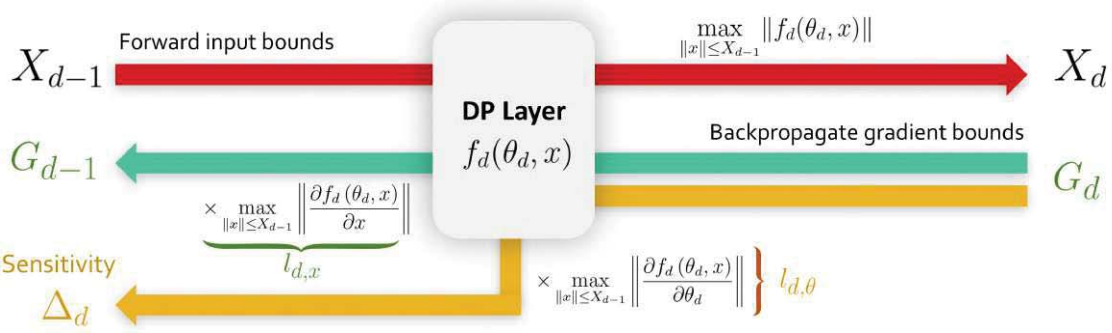


Figure 6.2: **Backpropagation for bounds** (Algorithm 8) computes the per-layer sensitivity Δ_d. The algorithm mimics backpropagation, with Vector-Jacobian products replaced by Scalar-Scalar products.

1. We extend the field of applications of Lipschitz-constrained neural networks. We extend the framework to **compute the Lipschitzness with respect to the parameters**. This general framework allows to track *layer-wise* sensitivities that depend on the loss and the model's structure. This is exposed in Section 6.2. We show that SGD training of deep neural networks can be achieved **without per-sample gradient clipping** using Lipschitz-constrained layers.
2. We establish connections between Gradient Norm Preserving (GNP) networks and improved privacy/utility trade-offs (Section 6.3.1). To the best of our knowledge, **we are the first ones to produce neural networks benefiting from both Lipschitz-based robustness certificates and privacy guarantees**.
3. Finally, a **Python package** companions the project, with pre-computed Lipschitz constants for each loss and each layer type. This is exposed in Section 6.3.2. It covers widely used architectures, including VGG, ResNets or MLP Mixers. Our package enables the use of larger networks and larger batch sizes, as illustrated by our experiments in Section 6.4.

6.2 Clipless DP-SGD with ℓ-Lipschitz networks

Our framework relies on the computation of the maximum gradient norm of a network w.r.t its parameters to obtain a *per-layer* sensitivity Δ_d. It is based on the recursive formulation of the chain rule involved in backpropagation and requires some natural assumptions that we highlight below.

Requirement 1 (Lipschitz loss.)**.** *The loss function $\hat{y} \mapsto \mathcal{L}(\hat{y}, y)$ must be L-Lipschitz with respect to the logits $\hat{y}$ for all ground truths $y \in \mathcal{Y}$. This is notably the case of Categorical Softmax-Crossentropy.*

The Lipschitz constants of common supervised losses has been computed and reported in the appendix.

Requirement 2 (Bounded input)**.** *There exists $X_0 > 0$ such that for all $x \in \mathcal{X}$ we have $\|x\| \leq X_0$. This is typically the case for images of height H, width W, and channels C. For pixel intensity in $[0, 1]$ we have that $X_0 \leq \sqrt{HWC}$. For tabular data, it is not uncommon to clip the extreme values.*

We recall that there exist two families of strategies to enforce Lipschitz constraints:

1. With a differentiable reparametrization $\Pi : \mathbb{R}^p \to \Theta$ where $\tilde{\theta} = \Pi(\theta)$: the weights $\tilde{\theta}$ are used during the forward pass, but the gradients are back-propagated to θ through Π. This turns the training into an unconstrained optimization problem on the landscape of the loss $\mathcal{L} \circ f \circ \Pi$.

2. With a suitable projection operator $\Pi : \mathbb{R}^p \to \Theta$: this is the celebrated Projected Gradient Descent (PGD) algorithm (Bubeck et al., 2015) applied on the landscape of the loss $\mathcal{L} \circ f$.

Option 1 requires the analysis of the Lipschitz constant of Π. If Θ is convex, then Π is 1-Lipschitz w.r.t the projection norm, otherwise the Lipschitz constant is generally unknown. For simplicity, option 2 will be the focus of this work.

Requirement 3 (Lipschitz projection)**.** *The Lipschitz constraints must be enforced with a projection operator $\Pi : \mathbb{R}^p \to \Theta$. This corresponds to Tensorflow* `constraints` *and Pytorch* `hooks`*. Projection is a post-processing (Dwork et al., 2006) of private data: it induces no privacy leakage.*

To compute the per-layer sensitivities, our framework mimics the backpropagation algorithm, where *Vector-Jacobian* products (VJP) are replaced by *Scalar-Scalar* products of element-wise bounds. For an arbitrary layer $x_d \mapsto f_d(\theta_d, x_d) := y_d$ the operation is sketched below, as a simple consequence of Cauchy-Schwartz inequality:

$$\underbrace{\nabla_{x_d}\mathcal{L} := (\nabla_{y_d}\mathcal{L})\frac{\partial f_d}{\partial x_d}}_{\text{Vector-Jacobian product: backpropagate gradients}} \implies \underbrace{\|\nabla_{x_d}\mathcal{L}\|_2 \leq \|\nabla_{y_d}\mathcal{L}\|_2 \times \left\|\frac{\partial f_d}{\partial x_d}\right\|_2}_{\text{Scalar-Scalar product: backpropagate bounds}}. \tag{6.3}$$

The notation $\|\cdot\|_2$ must be understood as the spectral norm for Jacobian matrices, and the Euclidean norm for gradient vectors. The scalar-scalar product is inexpensive. For Lipschitz layers, the spectral norm of the Jacobian $\|\frac{\partial f}{\partial x}\|$ is kept constant during training with projection operator Π. The bound of the gradient with respect to the parameters takes a simple form:

$$\|\nabla_{\theta_d}\mathcal{L}\|_2 \leq \|\nabla_{y_d}\mathcal{L}\|_2 \times \left\|\frac{\partial f_d}{\partial \theta_d}\right\|_2. \tag{6.4}$$

This term can be analytically bounded, as exposed in the following section.

6.2.1 Backpropagation for bounds

The pseudo-code of **Clipless DP-SGD** is sketched in Algorithm 9. The algorithm avoids per-sample clipping by computing a *per-layer* bound on the element-wise gradient norm. The computation of this *per-layer* bound is described by Algorithm 8 (graphically explained in Figure 6.2). Crucially, it requires to compute the spectral norm of the Jacobian of each layer with respect to input and parameters.

Input bound propagation (line 2). We compute $X_d = \max_{\|x\| \leq X_{d-1}} \|f_d(x)\|_2$. For activation functions it depends on their range. For linear layers, it depends on the spectral norm of the operator itself. This quantity can be computed through the SVD (Trefethen and Bau, 2022) or Power Iteration (Miyato et al., 2018), and constrained during training using projection operator Π. In particular, it covers the case of convolutions, for which tight bounds are known (Singla and Feizi, 2021a). For affine layers, it additionally depends on the magnitude of the bias $\|b_d\|$.

Remark 6.1. Tighter bounds in literature

Exact estimation of the Lipschitz bound is notoriously an NP-hard problem, as proven by (Virmaux and Scaman, 2018). Algorithm 9 can be seen as an extension of their *AutoLip* algorithm to the case of parameters, while their work focused on Lipschitness with respect to the input. Note that most methods discussed in section 2.1 can be used as a replacement for

Algorithm 9. Moreover, hybridizing our method with scalable certification methods can be a path for future extensions.

Computing maximum gradient norm (line 6). We now present how to bound the Jacobian $\frac{\partial f_d(\theta_d,x)}{\partial \theta_d}$. In neural networks, the parameterized layers $f(\theta, x)$ (fully connected, convolutions) are bilinear operators. Hence, we typically obtain bounds of the form:

$$\left\| \frac{\partial f_d(\theta_d, x)}{\partial \theta_d} \right\|_2 \leq K(f_d, \theta_d)\|x\|_2 \leq K(f_d, \theta_d) X_{d-1}, \tag{6.5}$$

where $K(f_d, \theta_d)$ is a constant that depends on the nature of the operator. X_{d-1} is obtained in line 2 with input bound propagation. Values of $K(f_d, \theta_d)$ for popular layers are reported in the appendix.

Backpropagate cotangent vector bounds (line 7). Finally, we bound the Jacobian $\frac{\partial f_d(\theta_d,x)}{\partial x}$. For activation functions, this value can be hard-coded, while for affine layers it is the spectral norm of the linear operator. Like before, this value is enforced by the projection operator Π.

Algorithm 8 Backpropagation for Bounds(f, X)

Input: Feed-forward architecture $f(\theta, \cdot) = f_D(\theta_D, \cdot) \circ \ldots \circ f_1(\theta_1, \cdot)$
Input: Weights $\theta = (\theta_1, \theta_2, \ldots \theta_D)$, input bound X_0

1: **for all** layers $1 \leq d \leq D$ **do**
2: $\quad X_d \leftarrow \max_{\|x\| \leq X_{d-1}} \|f_d(\theta_d, x)\|_2$. ▷ Input bounds propagation
3: **end for**
4: $G \leftarrow L/b$. ▷ Lipschitz constant of the (averaged) loss for batchsize b
5: **for all** layers $D \geq d \geq 1$ **do**
6: $\quad \Delta_d \leftarrow G \max_{\|x\| \leq X_{d-1}} \|\frac{\partial f_d(\theta_d,x)}{\partial \theta_d}\|_2$. ▷ Compute sensitivity from gradient norm
7: $\quad G \leftarrow G \max_{\|x\| \leq X_{d-1}} \|\frac{\partial f_d(\theta_d,x)}{\partial x}\|_2 = G l_d$. ▷ Backpropagate cotangent vector bounds
8: **end for**
9: **return** sensitivities $\Delta_1, \Delta_2 \ldots, \Delta_D$

Algorithm 9 Clipless DP-SGD with **per-layer** sensitivity accounting

Input: Feed-forward architecture $f(\theta, \cdot) = f_D(\theta_D, \cdot) \circ \ldots \circ f_1(\theta_1, \cdot)$
Input: Initial weights $\theta = (\theta_1, \theta_1, \ldots \theta_D)$, learning rate η, noise multiplier σ.

1: **repeat**
2: $\quad \Delta_1, \Delta_2 \ldots \Delta_D \leftarrow$ **Backpropagation for Bounds**(f, X).
3: $\quad$ Sample a batch $\mathcal{B} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_b, y_b)\}$.
4: $\quad$ Compute the averaged gradient for each layer d: $g_d := \frac{1}{b} \sum_{i=1}^{b} \nabla_{\theta_d} \mathcal{L}(f(\theta, x_i), y_i))$.
5: $\quad$ Sample per-layer noise: $\zeta_d \sim \mathcal{N}(\mathbf{0}, \sigma \Delta_d)$.
6: $\quad$ Perform perturbed gradient step: $\theta_d \leftarrow \theta_d - \eta(g_d + \zeta_d)$.
7: $\quad$ Enforce Lipschitz constraint with projection: $\theta_d \leftarrow \Pi(\theta_d)$.
8: $\quad$ Compute new (ϵ, δ)-DP guarantees with privacy accountant.
9: **until** privacy budget (ϵ, δ) has been reached.

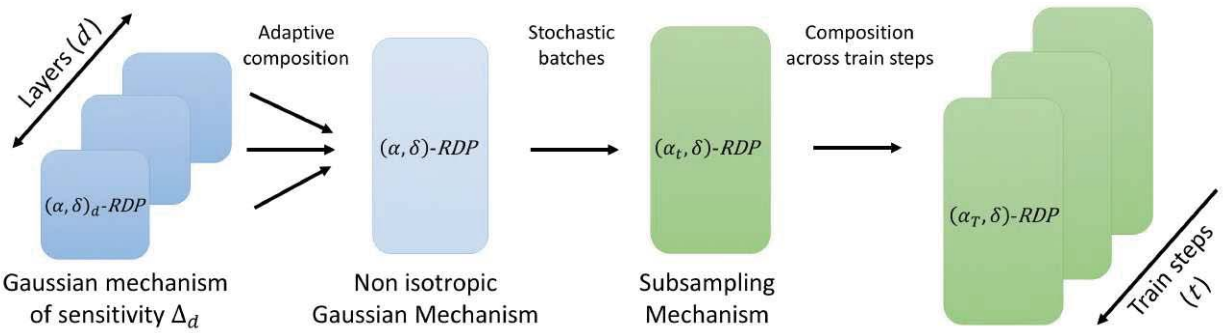


Figure 6.3: **Accountant for locally enforced differential privacy. (i)** The gradient query for each layer is made private using the Gaussian mechanism Dwork et al. (2014); **(ii)** their composition across the layers of the whole network can be seen as a non isotropic Gaussian mechanism, **(iii)** that benefits from amplification via sub-sampling Balle et al. (2018); **(iv)** the train steps are composed over the course of training.

Privacy accounting of Clipless DP-SGD

We keep track of (ϵ, δ)-DP values with a *privacy accountant* (Abadi et al., 2016), by composing different mechanisms. For a dataset with N records and a batch size b, it relies on two parameters: the sampling ratio $p = \frac{b}{N}$ and the "noise multiplier" σ defined as the ratio between effective noise strength ζ and sensitivity Δ. We propose two strategies to keep track of (ϵ, δ) values as the training progresses, based on either the "per-layer" sensitivities Δ_d (composition of Gaussian mechanisms), or by aggregating them into a "global" sensitivity $\Delta = \sqrt{\sum_d \Delta_d^2}$ (single isotropic Gaussian mechanism).

The "global" strategy. This strategy simply aggregates the individual sensitivities Δ_d of each layer to obtain the global sensitivity of the whole gradient vector $\Delta = \sqrt{\sum_d \Delta_d^2}$. The origin of the clipping-based version of this strategy can be traced back to McMahan et al. (2018). With noise variance $\sigma^2\Delta^2$ we recover the accountant that comes with DP-SGD. It tends to overestimate the true sensitivity (in particular for deep networks), but its implementation is straightforward with existing tools.

The "per-layer" strategy. Recall that we are able to characterize the sensitivity Δ_d of every layer of the network. Hence, we can apply a different noise to each of the gradients. We dissect the whole training procedure in Figure 6.3.

1. On each layer, we apply a Gaussian mechanism with noise variance $\sigma^2\Delta_d^2$.
2. Their composition yields an other Gaussian mechanism with non isotropic noise.
3. The Gaussian mechanism benefits from privacy amplification via subsampling (Balle et al., 2018) thanks to the stochasticity in the selection of batches of size $b = pN$.
4. Finally an epoch is defined as the composition of $T = \frac{1}{p}$ sub-sampled mechanisms.

At same noise multiplier σ, "per-layer" strategy tends to produce a higher value of ϵ per epoch than the "global" strategy, but has the advantage over the latter to add smaller effective noise ζ to each weight. Different layers exhibit different maximum gradient bounds - and in turn this implies different sensitivities. This also suggests that different noise multipliers σ_d can be used for each layer. This open extensions for future work.

We rely on the `autodp`[1] library (Wang et al., 2019b; Zhu and Wang, 2019, 2020) as it uses the Rényi Differential Privacy (RDP) adaptive composition theorem (Mironov, 2017; Mironov et al., 2019), that ensures tighter bounds than naive DP composition. Following standard practices of the community (Ponomareva et al., 2023), we used *sampling without replacement* at each epoch (by shuffling examples), but we reported ϵ assuming *Poisson sampling* to benefit from privacy amplification (Balle et al., 2018).

6.3 Signal-to-noise ratio analysis

We discuss how the tightness of the bound provided by Algorithm 8 can be controlled.

6.3.1 Theoretical analysis of Clipless DP-SGD

In some cases we can manually derive the bounds across diverse configurations.

Theorem (informal) 1. Gradient Norm of Lipschitz Networks. *Assume that every layer f_d is K-Lipschitz, i.e $l_1 = \cdots = l_D = K$. Assume that every bias is bounded by B. We further assume that each activation is centered in zero (i.e $f_d(\mathbf{0}) = \mathbf{0}$, like ReLU, tanh, GroupSort...). We recall that $\theta = [\theta_1, \theta_2, \dots \theta_D]$. Then the global upper bound of Algorithm 9 can be expanded analytically.*

1. ***If** $K < 1$ **we have:***

$$\|\nabla_\theta \mathcal{L}(f(\theta, x), y)\|_2 = \mathcal{O}\left(L\left(K^D(X_0 + B) + 1\right)\right). \tag{6.6}$$

Due to the $K^D \ll 1$ term this corresponds to a vanishing gradient phenomenon (Pascanu et al., 2013). The output of the network is essentially independent of its input, and training is nearly impossible.

2. ***If** $K > 1$ **we have:***

$$\|\nabla_\theta \mathcal{L}(f(\theta, x), y)\|_2 = \mathcal{O}\left(LK^D\left(X_0 + B + 1\right)\right). \tag{6.7}$$

Due to the $K^D \gg 1$ term this corresponds to an exploding gradient phenomenon (Bengio et al., 1994). The upper bound becomes vacuous for deep networks: the added noise ζ will be too high.

3. ***If** $K = 1$ **we have:***

$$\|\nabla_\theta \mathcal{L}(f(\theta, x), y)\|_2 = \mathcal{O}\left(L\left(\sqrt{D} + X_0\sqrt{D} + \sqrt{BX_0}D + BD^{3/2}\right)\right), \tag{6.8}$$

which for linear layers without biases further simplify to $\mathcal{O}(L\sqrt{D}(1 + X_0))$.

The formal statement can be found in appendix. However, setting $K = 1$ merely ensures that $\|\nabla_x f\| \leq 1$, and in the worst-case scenario we could have $\|\nabla_x f\| \ll 1$ almost everywhere. This results in a situation where the bound of case 3 in Theorem 1 is not tight, leading to an underfitting regime as in the case $K < 1$. With Gradient Norm Preserving (GNP) networks, we expect to mitigate this issue.

[1] `https://github.com/yuxiangw/autodp` distributed under Apache License 2.0.

Controlling K with Gradient Norm Preserving (GNP) networks. GNP (Li et al., 2019a) networks are 1-Lipschitz neural networks with the additional constraint that the Jacobian of layers consists of orthogonal matrices. They fulfill the Eikonal equation $\left\| \frac{\partial f_d(\theta_d, x_d)}{\partial x_d} \right\|_2 = 1$ for any intermediate activation $f_d(\theta_d, x_d)$. As a consequence, the gradient of the loss with respect to the parameters is bounded by

$$\|\nabla_{\theta_d} \mathcal{L}\| \leq \|\nabla_{y_D} \mathcal{L}\| \times \left\| \prod_{d<i\leq D} \frac{\partial f_i(\theta_i, x_i)}{\partial x_i} \right\| \times \left\| \frac{\partial f_d(\theta_d, x_d)}{\partial \theta_d} \right\| = \|\nabla_{y_D} \mathcal{L}\| \times \left\| \frac{\partial f_d(\theta_d, x_d)}{\partial \theta_d} \right\|, \tag{6.9}$$

which for weight matrices W_d further simplifies to $\|\nabla_{W_d} \mathcal{L}\| \leq \|\nabla_{y_D} \mathcal{L}\| \times \|f_{d-1}(\theta_{d-1}, x_{d-1})\|$. We see that this upper bound crucially depends on two terms than can be analyzed separately. On the one hand, $\|f_{d-1}(\theta_{d-1}, x_{d-1})\|$ depends on the scale of the input. On the other, $\|\nabla_{y_D} \mathcal{L}\|$ depends on the loss, the predictions and the training stage. We show below how to intervene on these two quantities.

Controlling X_0 with input pre-processing. The weight gradient norm $\|\nabla_{W_d} \mathcal{L}\|$ indirectly depends on the norm of the inputs. Multiple strategies are available to keep this norm under control: projection onto the ball ("norm clipping"), or projection onto the sphere ("normalization"). In the domain of natural images, this result sheds light on the importance of color space: RGB, HSV, etc. Empirically, a narrower distribution of input norms would make up for tighter gradient norm bounds.

Controlling L with the hybrid approach: loss gradient clipping As training progresses, the magnitude of $\|\nabla_f \mathcal{L}\|$ tends to diminish when approaching local minima, falling below the upper bound and diminishing the signal to noise ratio. Fortunately, for Lipschitz constrained network, the norm of the elementwise-gradient remains lower bounded throughout (Béthune et al., 2022). Since the noise amplitude only depends on the architecture and the loss, and remains fixed during training, the loss with the best signal-to-noise ratio would be a loss whose gradient norm w.r.t the logits remains constant during training. For the binary classification case, with labels $y \in \{-1, +1\}$, this yields the loss $\mathcal{L}_{KR}(\hat{y}, y) = -y\hat{y}$, that arises in Kantorovich-Rubinstein duality (Villani, 2008):

$$\mathcal{W}_1(P, Q) := \frac{1}{\ell} \inf_{f \in \ell\text{-Lip}(\mathcal{D}, \mathbb{R})} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}_{KR}(f(x), y)], \tag{6.10}$$

where $\mathcal{W}_1(P, Q)$ is the Wasserstein-1 distance between the two classes P and Q.

Another way to ensure a high signal-to-noise ratio is to diminish the noise and clip the gradients of the loss w.r.t the logits. We emphasize that *this is different from the clipping of the "parameter gradient"* $\nabla_\theta \mathcal{L}$ done in DP-SGD. Here, *any intermediate gradient* $\nabla_{f_d} \mathcal{L}$ can be clipped during backpropagation. This can be achieved with a special "*clipping layer*" that behaves like the identity function at the forward pass, and clips the gradient during the backward pass. In DP-SGD the clipping is applied on the element-wise gradient $\nabla_{W_d} \mathcal{L}$ of size $b \times h^2$ for matrix weight $W_d \in \mathbb{R}^{h\times h}$ and batch size b, and clipping it can cause memory issues or slowdowns (Lee and Kifer, 2021). In our case, $\nabla_{y_D} \mathcal{L}$ is of size $b \times h$: this is far smaller, especially for the last layer. Note that in this setting the Lipschitz condition does not hold anymore; instead, it obeys the more general "generalized Lipschitzness" condition introduced in Das et al. (2023). Moreover, this clipping is compatible with the adaptive clipping introduced by (Andrew et al., 2021): the quantiles of $\|\nabla_{y_D} \mathcal{L}\|$ can be privately estimated for a small privacy budget. This allows to effectively reduce the noise while ensuring that most of the gradients remain unbiased. Furthermore, this bias of this clipping can be characterized.

Proposition (informal) 1 (Bias of loss gradient clipping in binary classification tasks)**.** *Let $\mathcal{L}_{BCE}$ be the binary cross-entropy loss, with sigmoid activation. Assume that the loss gradient (w.r.t the logits) $\nabla_{\hat{y}}\mathbb{E}_{\mathcal{D}}[\mathcal{L}_{BCE}(\hat{y}, y)]$ is clipped to norm at most $C > 0$. Then there exists $C' > 0$ such that for all $C \leq C'$ a gradient descent step with the clipped gradient is identical in direction to the gradient descent step obtained from the loss $\mathcal{L}_{KR}(\hat{y}, y) = -\hat{y}y$.*

As observed in chapter 4 this descent direction yields classifiers with high certifiable robustness, but lower clean accuracy. Therefore, in practice, we set the adaptive clipping threshold at not less than the 90%-th quantile to mitigate the bias and avoid utility drop.

6.3.2 Lip-dp library

To foster and spread accessibility, we provide an open source TensorFlow library for Clipless DP-SGD training, named `lip-dp`, with Keras API. Its usage is illustrated in Figure 6.1.

The seminal work of Anil et al. (2019) proved that universal approximation in the set of ℓ-Lipschitz functions was achievable by this family of architectures. In practice, GNP networks are parametrized with GroupSort activation Anil et al. (2019); Tanielian and Biau (2021), Householder activation Mhammedi et al. (2017), and orthogonal weight matrices Li et al. (2019a,b).

6.4 Experimental results

We validate our implementation with a speed benchmark against competing approaches, and we present the privacy/utility Pareto fronts that can be obtained with GNP networks.

6.4.1 Evaluation of privacy, accuracy and robustness

For the comparisons, we leverage the DP-SGD implementation from Opacus. We perform a search over a broad range of hyper-parameter values: the configuration is given in Appendix B.3. We ignore the privacy loss that may be induced by this hyper-parameter search, which is a limitation per recent studies (Papernot and Steinke, 2022). The training is performed on a randomly initialized network, without data augmentation and without fine-tuning.

Accuracy and Privacy. We validate the performance of our approach on tabular data from Adbench suite (Han et al., 2022) using an MLP, and report the result in Table 6.4. For MNIST (Fig. 6.5a) and Fashion-MNIST (Fig. 6.5b) we use a Lipschitz LeNet-like architecture, while the Cifar-10 experiment (Fig. 6.5c) relies on a combination of VGG, ResNet, and MLP Mixer architectures.

Robustness and Privacy. One of the most prominent advantage of Lipchitz networks is their ability to provide robustness certificates against adversarial attacks, and was the primary motivation for their development (Szegedy et al., 2014; Li et al., 2019a; Fazlyab et al., 2019). For a ℓ-Lipschitz classifier f, with predictions $\bar{k} := \arg\max_k f_k(x)$ we recall that the decision is invariant under perturbation of norms smaller than $\frac{1}{\ell\sqrt{2}}(f_{\bar{k}}(x) - \arg\max_{i\neq\bar{k}} f_i(x))$. Therefore, for each perturbation radius r we can measure the accuracy of the perturbed network, and we report these results in Fig. 6.6. The computation of the certificates is straightforward, while methods applicable to conventional networks (like randomized smoothing (Cohen et al., 2019) or interval propagation, see remark 1), are expensive. This suggests that robust decisions and privacy are not necessarily antipodal objectives, contrary to what was observed in Song et al. (2019). The work of Wu et al. (2023) also study the link between certified robustness and privacy, albeit through the lens of adversarial training (Zhang et al., 2019).

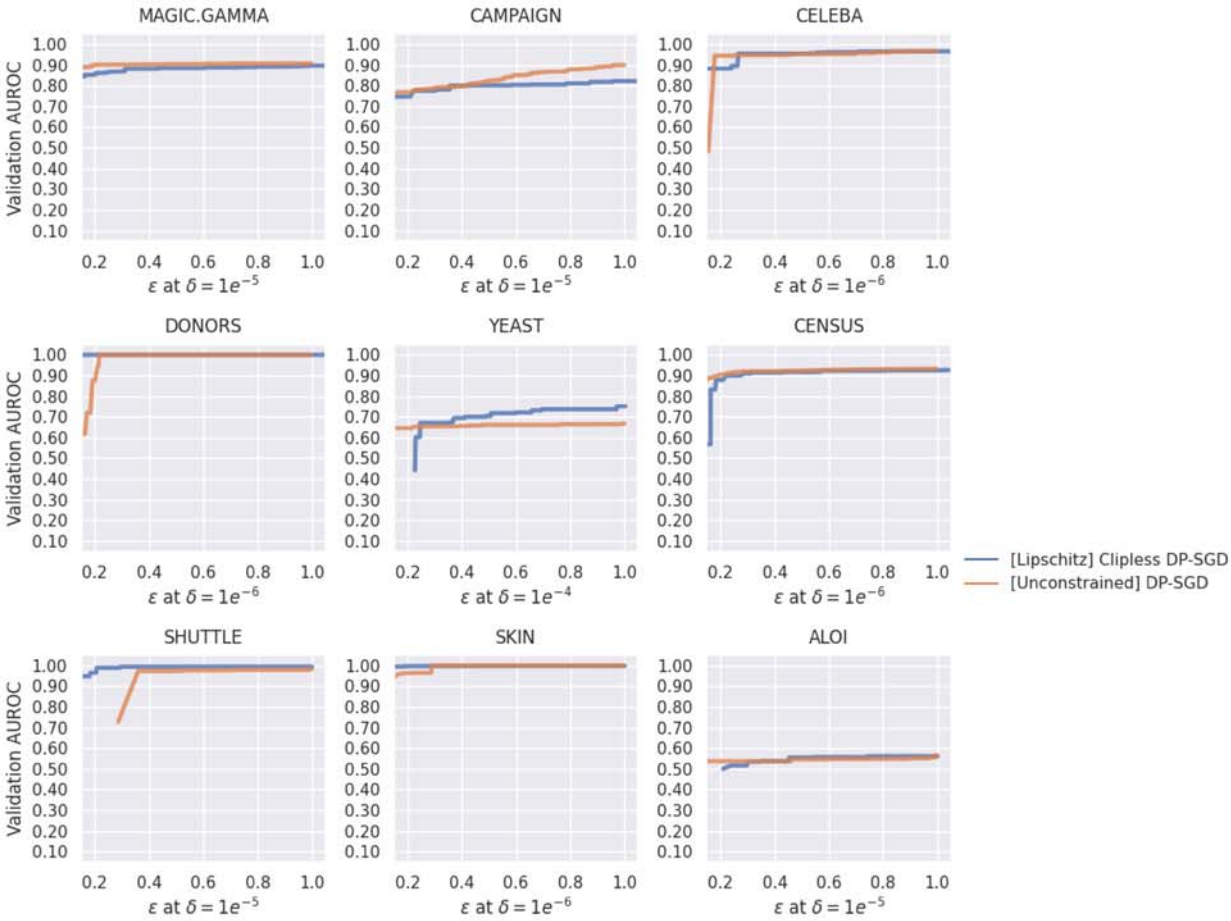
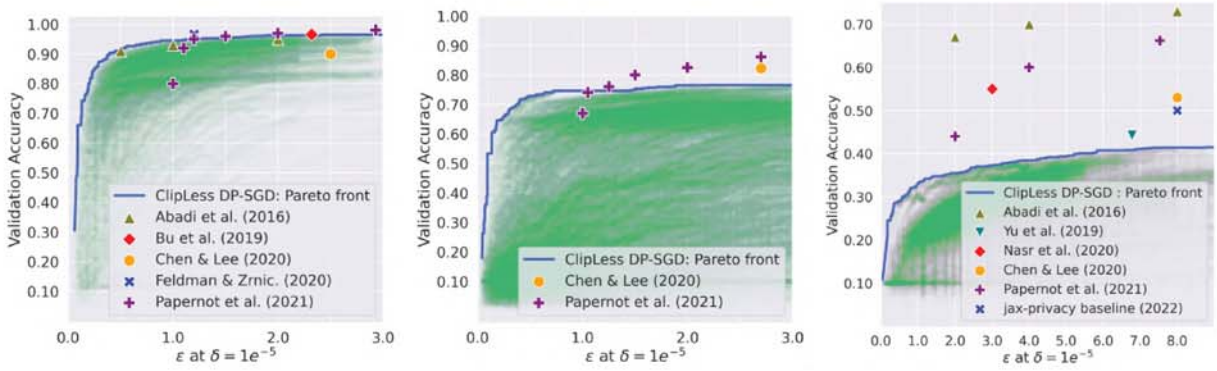


Figure 6.4: **Best AUROC values (in %) for models trained under (ϵ, δ)-DP privacy with $\epsilon = 1$**, on binary classification tasks of tabular data from Adbench datasets Han et al. (2022). We use a random stratified split into train (80%) / validation (20%).



(a) **MNIST**. (b) **F-MNIST**. (c) **CIFAR-10**.

Figure 6.5: **Our framework paints a clearer picture of the privacy/utility trade-off.** We trained models in an "out of the box setting" (no pre-training, no data augmentation and no handcrafted features) on multiple tasks. Each green dot corresponds to an (**accuracy**, ϵ) pair from an epoch of one of the runs, while the blue line is the Pareto front (convex hull of all the dots). While our results align with the baselines presented in other frameworks, we recognize the importance of domain-specific engineering. In this regard, we find the innovations introduced in Papernot et al. (2021); Tramer and Boneh (2021); De et al. (2022) and references therein highly relevant. These advancements demonstrate compatibility with our framework and hold potential for future integration.

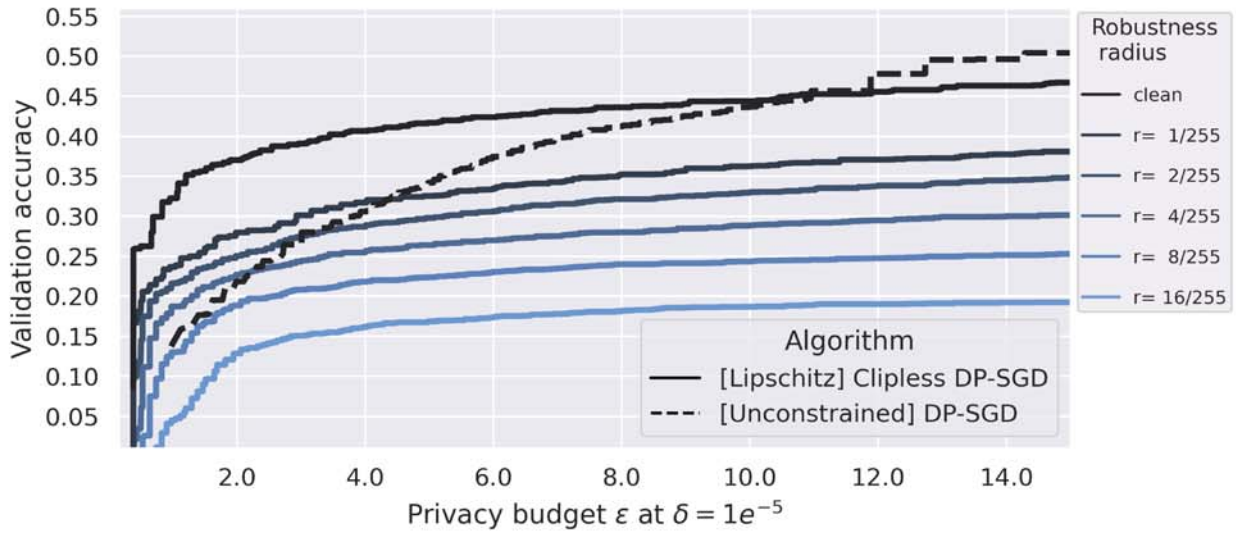


Figure 6.6: **Privacy/accuracy/robustness trade-off on Cifar-10:** We report the pareto front of robustness certificates at different radii r for Lipschitz constrained networks while unconstrained networks cannot produce robustness certificates. Models are trained in an "out of the box setting": no pre-training, no data augmentation and no handcrafted features.

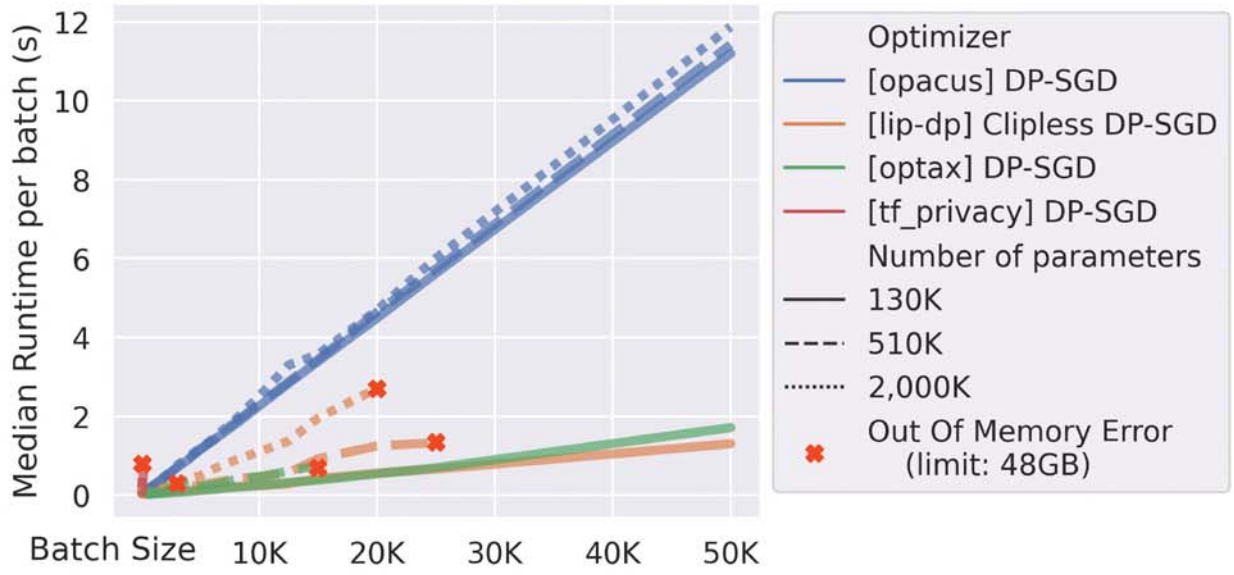


Figure 6.7: **Our approach outperforms concurrent frameworks in terms of runtime and memory:** we trained CNNs (ranging from 130K to 2M parameters) on CIFAR-10, and report the median batch processing time (including noise, and constraints application Π or gradient clipping).

6.4.2 Speed and memory consumption

We benchmarked the median runtime per batch of vanilla DP-SGD against the one of Clipless DP-SGD, on a CNN architecture and its Lipschitz equivalent respectively. The experiment was run on a GPU with 48GB video memory. We compare against the implementation of `tf_privacy`, `opacus` and `optax`. In order to allow a fair comparison, when evaluating Opacus, we reported the runtime with respect to the logical batch size, while capping the physical batch size to avoid Out Of Memory error (OOM). Although our library does not implement logical batching yet, it is fully compatible with this feature.

An advantage of the projection Π over per-sample gradient clipping is that its cost is independent of the batch size. Fig 6.7 validates that our method scales much better than vanilla DP-SGD, and is compatible with large batch sizes. It offers several advantages: firstly, a larger batch size contributes to a decrease of the sensitivity $\Delta \propto 1/b$, which diminishes the ratio between noise and gradient norm. Secondly, as the batch size b increases, the variance decreases at the parametric rate $\mathcal{O}(\sqrt{b})$, aligning with expectations. This observation does not apply to DP-SGD: clipping biases the direction of the average gradient, as noticed by Chen et al. (2020).

Corollary 4. Concentration of stochastic gradient around its mean. *Assume the samples (x, y) are i.i.d and sampled from an arbitrary distribution $\mathcal{D}$. We introduce the R.V $g = \nabla_\theta \mathcal{L}(x, y)$ which is a function of the sample (x, y), and its expectation $\bar{g} = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\nabla_\theta \mathcal{L}(x, y)]$. Then for all $u \geq \frac{2}{\sqrt{b}}$ the following inequality hold:*

$$\mathbb{P}(\|\frac{1}{b}\sum_{i=1}^{b} g_i - \bar{g}\| > uK) \leq \exp\left(-\frac{\sqrt{b}}{8}(u - \frac{2}{\sqrt{b}})^2\right). \tag{6.11}$$

Proof. The result is an immediate consequence of Example 6.3 p167 in Boucheron et al. (2013). We apply the theorem with the centered variable $X_i = \frac{1}{b}(g_i - \bar{g})$ that fulfills condition $\|X_i\| \leq \frac{c_i}{2}$ with

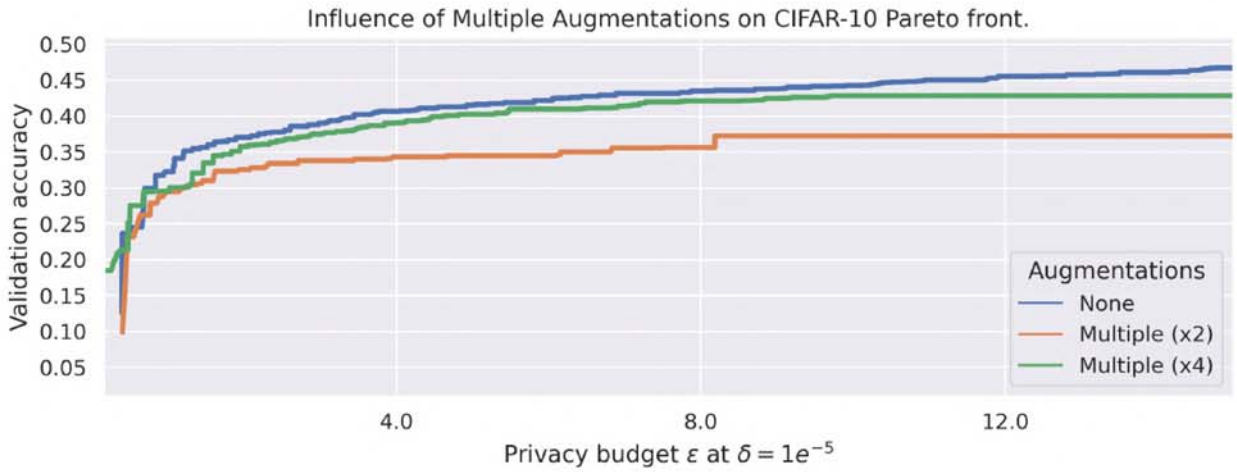


Figure 6.8: **Pareto front on Cifar-10 with the multiple augmentations trick of De et al. (2022)**. It appears that the increased diversity of images does not improve the validation accuracy. Maybe it is because the network *is not* in an un

$c_i = \frac{4K}{b}$ since $\|g_i\| \leq K$. Then for every $t \geq \frac{2K}{\sqrt{b}}$ we have:

$$\mathbb{P}(\|\frac{1}{b}\sum_{i=1}^{b} g_i - \bar{g}\| > t) \leq \exp\left(-\frac{\sqrt{b}}{8K^2}(t - \frac{2K}{\sqrt{b}})^2\right). \tag{6.12}$$

We conclude with the change of variables $u = \frac{t}{K}$. □

6.4.3 Compatibility with litterature improvements

Many method from the state of the art are based on improvement over the DP-SGD baseline. In this section we will review these improvements and check the compatibility with our approach.

Multiple augmentations We tested adding "multiple augmentation" introduced by De et al. (2022) on our approach but it did not yield improvements (see Appendix 6.8). This is probably due to the fact that we train our architecture from scratch: doing so require an architecture that requires a minimum amount of steps to converge. Such architecture are too small and not able to learn an augmented dataset as those are more prone to under-fitting. Another reason is that in vanilla DP-SGD the elementwise gradients of each augmentation can be averaged *before* the clipping operation, which has no consequences on the overall sensitivity of the gradient step. Whereas in our case, the average is computed *after* the loss gradient clipping.

Fine tuning a pretrained backbone: Clipless DP-SGD can work with a pre-trained backbone, however the fine tuned layers must be 1-Lipschitz. This can yield two approaches:

- **Using a 1-Lipschitz backbone:** We can then fine tune the whole network and benefits from pre-training. Unfortunately there is no 1-Lipschitz backbone available for the moment.
- **Using an unconstrained backbone:** Our approach can work with an unconstrained feature extractor using the following protocol: 1. the n last layers are dropped and replaced with a 1-Lipschitz classification network. 2. An input clipping layer is added at the beginning of the

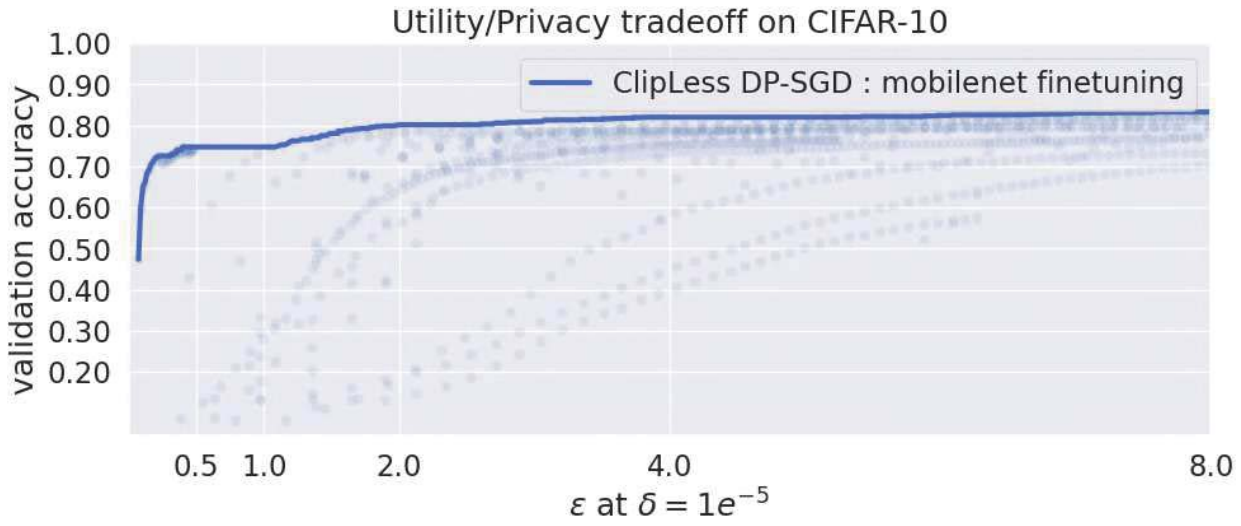


Figure 6.9: **Finetuning of a MobilenetV2 with Clipless DP-SGD.** A Lipschitz MLP (2 layers, and GroupSort activation) is trained using the backbone as a feature extractor. The backbone is not fine-tuned as it is not Lipschitz. Therefore a plateau is attained, depending on the quality of the feature extractor.

classification network to ensure that the inputs are bounded. This approach were tested using a MobilenetV2 backbone. Results are reported in fig 6.9.

6.5 Conclusion

We discuss here some of the limitations of the method, notably the bias induced by Lipschitz networks. We also compare Clipless DP-SGD to recent and efficient methods for clipping.

6.5.1 Bias of Clipless DP-SGD

Clipless DP-SGD *also* exhibits a bias, but this bias takes a different form than the bias induced by clipping in DP-SGD.

The bias of the optimizer. In DP-SGD (with clipping) the average gradient is biased by the elementwise clipping. Therefore, the clipping may slow down convergence or lead to sub-optimal solutions.

The bias of the model in the space in Lipschitz networks. This is a bias of the model, not of the optimizer. It has been shown that any classification task could be solved with a 1-Lipschitz classifier (Béthune et al., 2022), and in this sense, the bias induced by the space of 1-Lipschitz functions is not too severe. Better, this bias is precisely what allows to produce robustness certificates, see for example Yang et al. (2020).

Finally, there is the implicit bias. It is induced by a given architecture on the optimizer, which can have strong effects on effective generalization. For neural networks (Lipschitz or not), this implicit bias is not fully understood yet. But even on large Lipschitz models, it seems that the

	Instantiating per-sample gradient	Storing every layer's gradient	Instantiating non-DP gradient	Number of back propagations	Overhead independent of batch size
non-DP	✗	✗	✓	1	✓
TF-Privacy, like Abadi et al. (2016)	✓	✗	✓	B	✗
Opacus (Yousefpour et al., 2021)	✓	✓	✓	1	✗
FastGradClip (Lee and Kifer, 2021)	✓	✗	✗	2	✗
GhostClip (Li et al., 2021; Bu et al., 2022a)	✗	✗	✓	1	✗
Book-Keeping (Bu et al., 2023)	✗	✗	✗	1	✗
Clipless w/ Lipschitz (ours)	✗	✗	✓	1	✓
Clipless w/ GNP (ours)	✗	✗	✓	1	✓

Table 6.1: **Comparison of DP-SGD against existing techniques of literature**. Clipless DP-SGD is the only technique whose time/memory overhead depends exclusively on weight matrices but not the batch size.

	Time Complexity Overhead	Memory Overhead
non-DP	0	0
TF-Privacy, like Abadi et al. (2016)	$\mathcal{O}(BTpd)$	0
Opacus (Yousefpour et al., 2021)	$\mathcal{O}(BTpd)$	$\mathcal{O}(Bpd)$
FastGradClip (Lee and Kifer, 2021)	$\mathcal{O}(BTpd)$	$\mathcal{O}(Bpd)$
GhostClip (Li et al., 2021; Bu et al., 2022a)	$\mathcal{O}(BTpd + BT^2)$	$\mathcal{O}(BT^2)$
Book-Keeping (Bu et al., 2023)	$\mathcal{O}(BTpd)$	$\mathcal{O}(B\min(pd, T^2))$
Clipless w/ Lipschitz (ours)	$\mathcal{O}(Upd)$	0
Clipless w/ GNP (ours)	$\mathcal{O}(Upd + Vpd\min(p, d))$	0

Table 6.2: **Time and memory costs for each method** for feedforward networks. We assume weight matrices of shape $p \times d$, and a *physical* batch size B. For images, T is height$\times$width. U is the number of iterations in Power Iteration algorithm, and V the number of iterations in Björck projection. Typically $U, V < 15$ in practice. The overhead is taken relatively to non-DP training *without* clipping. This table is largely inspired from Bu et al. (2023).

Lipschitz constraint biases the network toward better robustness radii but worse clean accuracy, as frequently observed in the relevant literature.

Therefore, these biases influence the learning process differently, and they constitute two distinct (not necessarily exclusive) approaches. We illustrate the difference between these paradigms in Figure 6.10.

6.5.2 Efficiency of gradient clipping

Beyond the conventional implementation of gradient clipping that can be found in Opacus or Tf-privacy, recent developments proposed more efficient forms of clipping, or to get rid of the clipping introduced by Abadi et al. (2016) and to use renormalization instead.

In this regard, we can mention the works of Bu et al. (2022b) or Yang et al. (2022) that study alternative implementations of clipping based on renormalization, which eliminates the need to tune the clipping value, with convergence guarantees.

Other works study the alternative implementation of elementwise clipping to reduce the computational cost, like Bu et al. (2022a), Li et al. (2021) He et al. (2022), and Bu et al. (2023). Taking inspiration from Bu et al. (2023) we summarize each one in Tables 6.1 and 6.2.

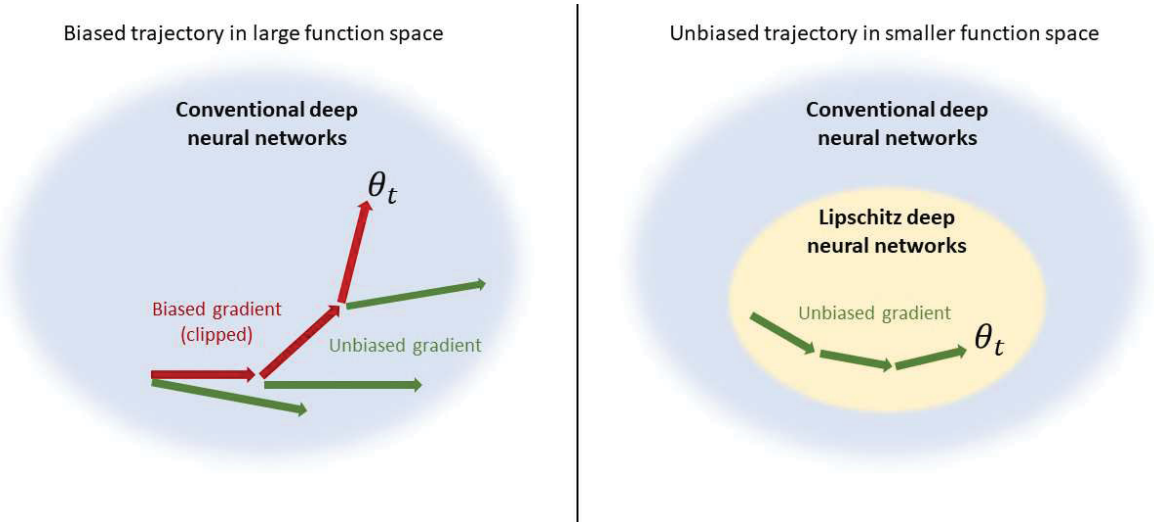


Figure 6.10: **Comparison between the bias of DP-SGD *with* clipping, and Clipless DP-SGD**. One is an instance of *optimizer bias*, and the other of *function space bias*.

6.5.3 Limitations

We measure the theoretical bounds with our framework at the start of training in Fig. 6.11a, and at the end of training in Fig. 6.11b. We see that the last layer benefits from bounds that are quite tight (around 30% to 50%) whereas the tightness drops for the deeper layers in MLP blocks (less than 10%). The explanation is that the bound

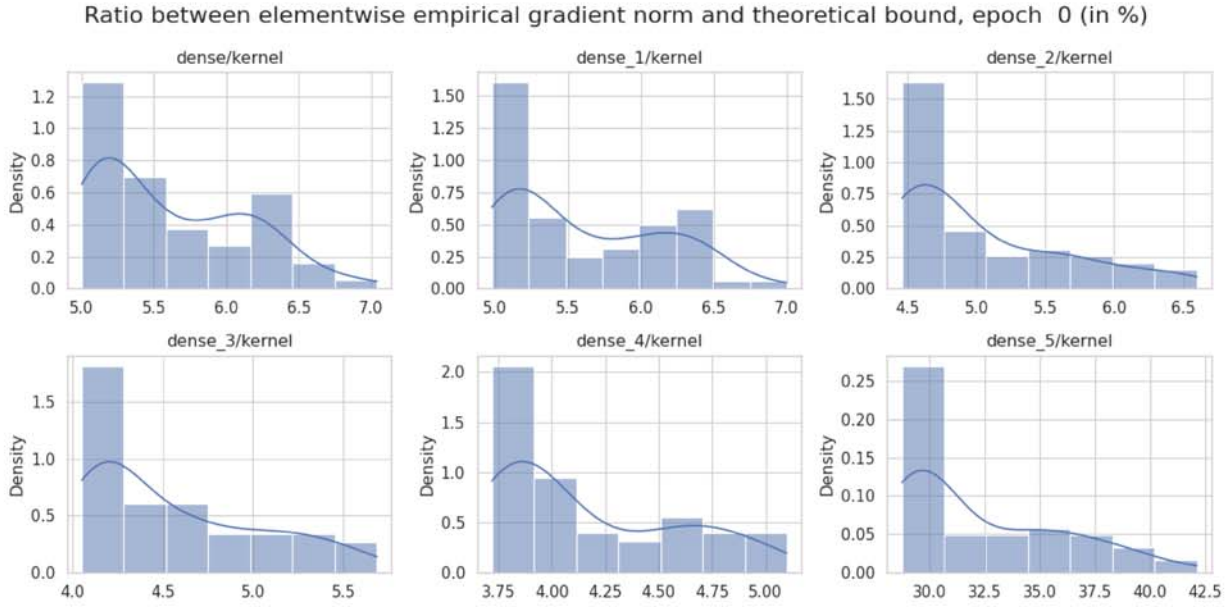
$$\left\| \frac{\partial f_d(\theta_d, x)}{\partial \theta_d} \right\|_2 \leq K(f_d, \theta_d) X_{d-1}, \tag{6.13}$$

might be overly pessimistic, because of the X_{d-1} term, and especially since Cauchy-Schwartz inequality is a "best-case bound" (i.e assuming that all vector involved in the inner product are co-linear), and do not account for the possible orthogonality between x_d and the cotangent vector. Nonetheless, with a ratio of 4%, we see that the noise and the gradient norm are of similar amplitude as soon as the batch size exceed $1/0.04 = 25$ examples, which is clearly our case with batch sizes that easily exceed $2,500$ on Cifar-10.

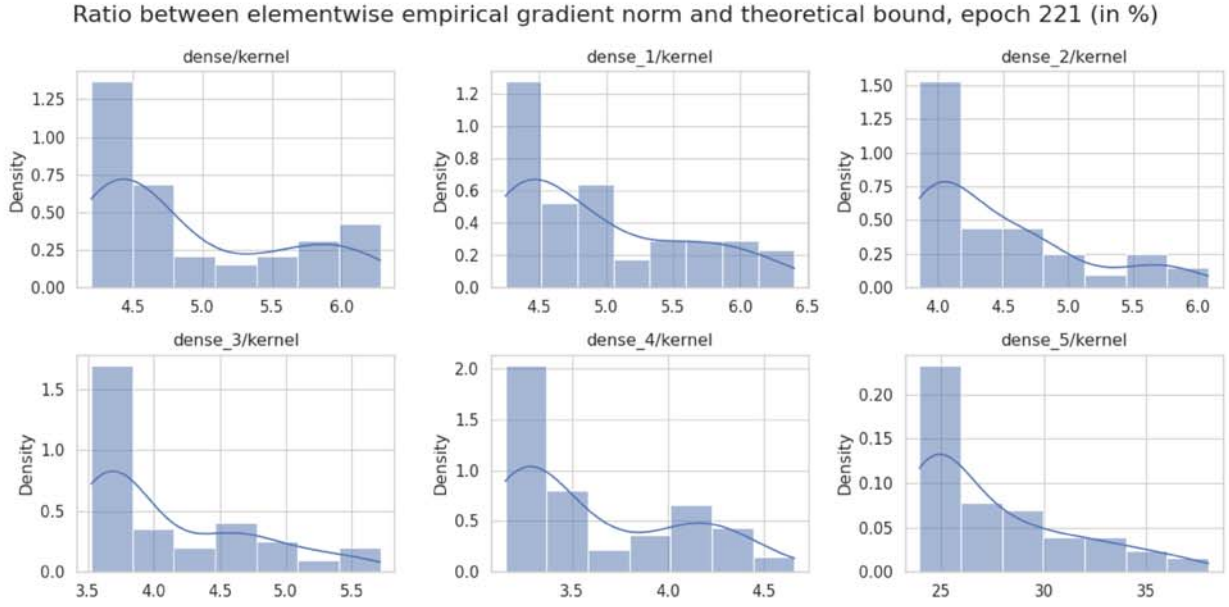
6.5.4 Future works and broader impact

Our framework offers a novel approach to address differentially private training, but also introduces new challenges. We primarily rely on GNP networks, where high-performing architectures are quite different from the usual CNN architectures. This serves as **(1) a motivation to further develop Gradient Norm Preserving architectures**. We anticipate that progress in these areas will greatly enhance the effectiveness of our approach. Additionally, to meet requirement 3, we rely on projections, necessitating additional efforts to incorporate recent advancements associated with differentiable reparametrizations Trockman and Kolter (2021); Singla and Feizi (2021b). Finally, as mentioned in Remark 1, our propagation bound method can be refined.

Beyond the application of Algorithms 8 and 9, our framework provides numerous opportunities to enhance our understanding of prevalent techniques identified in the literature. An in-depth exploration of these is beyond the scope of this work, so we focus on giving insights on promising tracks based on our theoretical analysis. Furthermore, the development of networks with known



(a) **MLP Mixer on Cifar-10, first epoch.**



(b) **MLP Mixer on Cifar-10, last epoch.**

Figure 6.11: **Comparison of histograms for different epochs.** We see that the ratios between empirical gradient norms and their upper bound tend to diminish with time, which is expected as the loss decreases toward a minimum. However, the effect remains small: as seen in chapter 4 the loss remains lower bounded above zero for Lipschitz networks.

Lipschitz constant with respect to parameters is a question of independent interest, making `lip-dp` **(2) a useful tool for the study of the optimization dynamics** in Lipschitz constrained neural networks.

Chapter 7

Gradient of convex potential fields and neural Monge maps

Lipschitz networks have key connections to optimal transport, thanks to Kantorovich-Rubinstein duality, and $\mathcal{W}_1$ distance. Similarly, convexity constraints arise naturally in the context of $\mathcal{W}_2$ distance, which benefits from numerous advantages over $\mathcal{W}_1$. Indeed, $\mathcal{W}_2$ distance can be formulated as the following optimization problem (Korotin et al., 2021):

$$\mathcal{W}_2(P, Q) = \sup_{\phi \in \text{Convex}(\mathcal{X}, \mathbb{R})} \mathbb{E}_{x \sim P}[\phi(x)] + \mathbb{E}_{z \sim Q}[\phi^c(z)], \tag{7.1}$$

where ϕ^c is the c-transform of ϕ defined as $\phi^c(x) := \min_y \frac{1}{2}\|x - y\| - \phi(y)$. Interestingly, the optimal Monge map T is guaranteed to be the gradient of some convex function (Gangbo and McCann, 1996):

$$T_\theta(x) := \nabla_x f_\theta(x). \tag{7.2}$$

However, the set of convex functions is much harder to parametrize than Lipschitz functions. In this chapter, we do a short literature on the field, and we discuss the main challenges it faces.

1. **First**, we give a general "cookbook" to parametrize convex functions, following the approach of Amos et al. (2017). We also discuss some of the difficulties of parametrizing convex functions.
2. **Then**, we discuss the possibility of parametrizing T *directly* as a gradient without relying on Autodiff.
3. **Next**, we illustrate the usage of convex networks in various topics related to optimal transport, such as center-outward distribution or fairness with counterfactual examples.
4. **Finally**, we propose a parametric estimator that may be used for multivariate quantile regression with some statistical guarantees.

The preliminary results of this chapter are mostly unpublished.

Contents

7.1 Parametrization of convex functions

In this section, we discuss the parametrization of convex neural networks.

7.1.1 Input Convex Neural Networks (ICNN)

The training of convex networks (w.r.t the input x, not the parameters) is consistently proven to be challenging (Korotin et al., 2021). Notably, to date, there is a lack of consensus regarding the optimal architecture.

Convex networks in literature. All architectures typically stem from the original proposal of Amos et al. (2017). This seminal work introduces non-negative weights in affine layers and non-decreasing non-linearities. In Warin (2023) the GroupMax activation is suggested. In Bunne et al. (2022) the use of a quadratic potential is recommended. The work of Korotin et al. (2021) is a comprehensive benchmark of the contemporary technique for $\mathcal{W}_2$ distance estimation, that also covers the question of the architecture.

Vanishing gradient issues Convex functions are generally not stable under composition, and obey instead a set of more restrictive rules (see example 1), which makes their parametrization more complex than the set of 1-Lipschitz functions.

Exemple 7.1. Non stability of convex functions under composition.

The canonical example is given by $f(x) = x^2$ and $g(x) = \exp(-x)$. Both functions can be easily checked to be convex, but their composition $g \circ f$ is obviously not convex since $\exp(-x^2)$ yields a "bell curve". Here, the issue stems from the fact that g is decreasing.

We detail below a set of common tools used for the parametrization of convex functions.

Remark 7.1. Convex Networks Cookbook.

Let $(f_i : \mathbb{R}^d \to \mathbb{R})_{1 \leq i \leq n}$ be a set of n convex functions, and we note $f : \mathbb{R}^d \to [f_1(x), \ldots, f_n(x)]$ the vectorized function. The following tools are available:

1. **Affine.** Let $P \in \mathbb{R}^{d' \times d}$ be any matrix, and $b \in \mathbb{R}^{d'}$. Then $x \mapsto Px + b$ is convex.
2. **Quadratic potential.** This component has been proposed in Bunne et al. (2022). Let $M \in \mathbb{R}^{d \times d}$ be any positive semi-definite matrix. Then $x \mapsto x^T M x$ is convex. In particular $\|P(x-b)\|_2^2$ is convex w.r.t x. Note that the gradient of a quadratic potential

is a linear form $M(x-m)$. In particular, the gradient is linear. A single-layer ICNN built with this potential yields linear Monge maps.

3. **Composition.** Let $g : \mathbb{R} \to \mathbb{R}$ be a non-decreasing convex function. Then $g \circ f_i$ is convex. Common choices of non-decreasing functions g are given below.
4. **Maximum.** Notoriously, the maximum $x \mapsto \max_i f_i(x)$ is convex. Indeed, the epigraph of the maximum function is the intersection of epigraphs of the f_i's, and an intersection of convex sets is convex.
5. **Log-sum-exp**, aka "smooth maximum". The function $x \mapsto \tau \log \sum_i \exp\left(f_i(x)/\tau\right)$ is convex. In the limit $\tau \to 0$ we retrieve the max function, as above.
6. **Affine positive composition.** Let $W_+ \in \mathbb{R}_+^n$ be a matrix with non-negative weights. Then $x \mapsto W_+^T f(x)$ is convex. Observe that $\frac{W_+}{\|W_+\|_1}$ is a probability vector.

Property 1 is used for skip connections from input to intermediate layers. Property 2 is usually used in the first layer. Property 3 is used for activation functions. Properties 4 and 5 are also useful as non-linearities.

Activation function. The activation function must be non-decreasing. Optionally, we want its Jacobian to be well conditioned: $\sigma_{\max}$ must not be too big, and the stable rank must be as high as possible (see the definition in Equation 7.3).

Parametrization of non-negative weights. The literature usually relies on ReLU $\Pi(W) = \max(0, W)$ or Softplus $\Pi(x) = \ln 1 + \exp x$ to ensure that the re-parametrized weights $\tilde{W} = \Pi(W)$ are non negative. This is currently unseen in literature, but absolute $\Pi_{||}(x) = |x|$ or quadratic activation $\Pi_2(x) = x^2$ are also possible. Observe that if M is an orthogonal matrix, its elementwise square $\Pi_2(M)$ is *orthostochastic* (Brualdi, 2006): not only the elements are positive, the rows and the columns sum to one. Therefore, the tools presented in Chapter 2 can be used to parametrize ortho-stochastic matrices.

Remark 7.2. Softplus and straight-through estimator.

One of the approaches consists in finding a *diffeomorphism* between $\mathbb{R}_+^{n\times n}$ and $\mathbb{R}^{n\times n}$. In this regard, Softplus with *straight through estimator trick* (Bengio et al., 2013) enjoys interesting properties. In the forward pass, Π behaves like Softplus, and in the backward pass, Π behaves like the identity function. This mitigates the vanishing gradient phenomenon. Furthermore, as for many diffeomorphisms with straight-through tricks, it is possible to interpret the descent step as a Riemannian gradient step in some curved space whose geometry depends on Π (see Niculae (2020) for more details). Taking this curvature into account is precisely what characterizes order-2 methods, and as such, the straight-through estimator can be a cheap way to run an order-2 optimization method with order-1 gradients, if the map Π fulfills the right properties.

7.1.2 Stable rank of positive matrices

The last property 6 is used for affine transformations in intermediate layers. This causes some difficulties: the spectrum of non-negative matrices may be ill-conditioned (see figures 7.1). High singular values can elicit *exploding gradient* phenomenon. This can be avoided with re-normalization by $\sigma_{\max}$ - however in this case we might face vanishing gradient issues when the other σ_i's are too

small. We propose to measure the behavior of the transformation $x \mapsto f(x)$ through the *stable rank* (Rudelson and Vershynin, 2007) (sometimes called "*numerical rank*") of its Jacobian matrix evaluated in x:

$$\mathfrak{r}(f, x) := \frac{\|\partial_x f\|_F^2}{\|\partial_x f\|_2^2} = \sum_{i=1}^{n} \left(\frac{\sigma_i}{\sigma_{\max}} \right)^2 \leq \operatorname{rank}(\partial_x f). \tag{7.3}$$

This metric is an alternative to the algebraic rank and has proven to be more relevant for numerical applications (Sanyal et al., 2020). Indeed, when a singular value is very close to zero, by all means, the associated transformation M behaves (numerically) like a rank-deficient transformation. While the stable rank measures an "average" behavior it is important to also monitor the value of $\sigma_{\max}$: this corresponds to a best-case scenario for the gradient norm when it is back-propagated through successive layers. This single metric is a useful proxy to summarize the "spreadness" of the whole singular values histogram. The stable rank is even linked to generalization results in deep neural networks (Bartlett et al., 2017; Neyshabur et al., 2018). The goal is to find parametrizations (and initializations) that ensure that the singular value spectrum is not degenerated.

Initialization of non-negative weights. Ensuring the good properties of Π derivative is not sufficient: the initialization of weights $\tilde{W}$ plays an important role. Once again, vanishing and exploding gradient issues must be avoided. Following guidelines of Glorot and Bengio (2010), the norm of feature vectors should remain invariant under width scaling. In their seminal work, the standard deviation of the weights is chosen such that $y_i \sim \mathcal{N}(0, \frac{1}{\sqrt{n}})$ for $y = Wx$. The question of initialization is also discussed in Bunne et al. (2022) where different strategies are proposed. In the first one, the Monge map behaves like the identity function, and in the second one like a Monge map between (mono-modal) Gaussian approximations of P and Q. Both initializations are higher quality than random, in the sense that the random Monge map T verifies $T_{\#}P \subset (\operatorname{dom} P \cup \operatorname{dom} Q)$, which ensures that the problem is not ill-conditionned. However, this does not solve the issues related to the spectrum of the Jacobians.

Exemple 7.2. Spectrum of positive weights matrices.

In Figure 7.1 we explore the effect of different initialization schemes on the singular values, on 512×512 matrices. For faithful comparison, every weight matrix W is re-scaled such that $\|W\|_2 := \sigma_{\max} = 1$. We explore the following settings:

1. **Glorot-Uniform** Glorot and Bengio (2010) corresponds to $x \sim \mathcal{U}([-a, a])$ with $a = \sqrt{\frac{6}{d_1+d_2}}$ is the default scheme proposed in deep learning framework. The weights **are not** positive: this experience is for comparison only.
2. **Softplus** $\log(1 + \exp(x))$ with $x \sim \mathcal{U}([-a, a])$, positive with a degenerate spectrum.
3. **ReLU** $\max(0, x)$ with $x \sim \mathcal{U}([-a, a])$, positive with a degenerate spectrum.
4. **Abs** $\|x\|$ with $x \sim \mathcal{U}([-a, a])$, positive with a degenerate spectrum. This is equivalent to sampling in the positive quadrant $\mathcal{U}([0, a])$.
5. **Softmax** $\frac{\exp x_i}{\sum_j \exp x_j}$ with $x \sim \mathcal{U}([-a, a])$, positive with a somewhat spread spectrum. Note that the Softmax reduction is performed either on rows or columns, but yields the same spectrum for square matrices. This yields a left or right stochastic matrix.
6. **Huber** $\frac{1}{2}x_i^2$ if $|x_i| \leq 1$, and $|x_i| - \frac{1}{2}$ otherwise, with $x \sim \mathcal{U}([-a, a])$, positive with a degenerate spectrum.
7. **Orthostochastic** matrices form a strict subset of the family of bi-stochastic matrices. It is defined as the elementwise square of an orthogonal matrix. The spectrum is also

degenerated.

8. **Softmax+Orthogonal**. Here a softmax transformation is applied to an orthogonal matrix. The spectrum is even more spread.

We also compute the stable rank $\mathfrak{r}(f, x)$ at initialization. This confirms that most initializations yield layers that behave like rank 1, rank 2 or rank 3 matrices. We also report the ratio between the numerical rank and the maximum possible rank (512).

Initialization	$\mathfrak{r}(f,x)$	$\mathfrak{r}(f,x)/512$
Glorot-Uniform	130.8	25.5%
Softplus+Glorot-Uniform	1	0.20%
ReLU+Glorot-Uniform	2.64	0.51%
Abs+Glorot-Uniform	1.33	0.26%
Softmax+Glorot-Uniform	36.8	7.18%
Huber+Glorot-Uniform	1.8	0.35%
Orthostochastic	3.0	0.58%
Softmax+Orthogonal	87.3	17.0%

We see that Glorot-Uniform exhibit the best spectrum (as expected) but fail to fulfill the basic requirement of positive weights. The second best scheme seems to be the *softmax* transformation applied on an orthogonal matrix. Grid-Search over those initializations revealed that this initialization scheme was the easiest to optimize with, on various $\mathcal{W}_2$ estimation tasks. **This experiment also reveals that the default choices of literature (ReLU and softplus) are typically not the best with regard to this criterion.**

7.2 Direct parametrization of convex gradients

We are primarily interested in the transportation plan $T = \nabla_x f$, where the potential f is only necessary at training time in certain formulations of the problem, such as in the works of Korotin et al. (2020, 2021). This has some drawbacks:

- It is necessary to parametrize and optimize over the set of convex functions, which has the disadvantages presented in the last section.
- This requires the use of Autodiff to retrieve $\nabla_x f$ which increases runtime and memory consumption.

7.2.1 Parametrization of gradients

This raises the question of whether or not the gradient of convex functions can be parameterized directly. A first attempt has been made in the work of Richter-Powell et al. (2021) but it does not scale to more than 2 layers deep. Parametrizing gradients alone, without relying on Autodiff, is itself a complicated task. An efficient parametrization would have tremendous consequences that go beyond optimal transport, and would be of huge interest in the context of ODE solving or physic-informed neural networks (PINN), see Richter-Powell et al. (2022) for example.

Property 8 (Symmetry of second derivatives.)**.** *Per Schwartz's theorem Widder (2012), if $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable, and if its second derivative is continuous, then the Hessian matrix of f is symmetric:*

$$(H_x f) = (H_x f)^T . \tag{7.4}$$

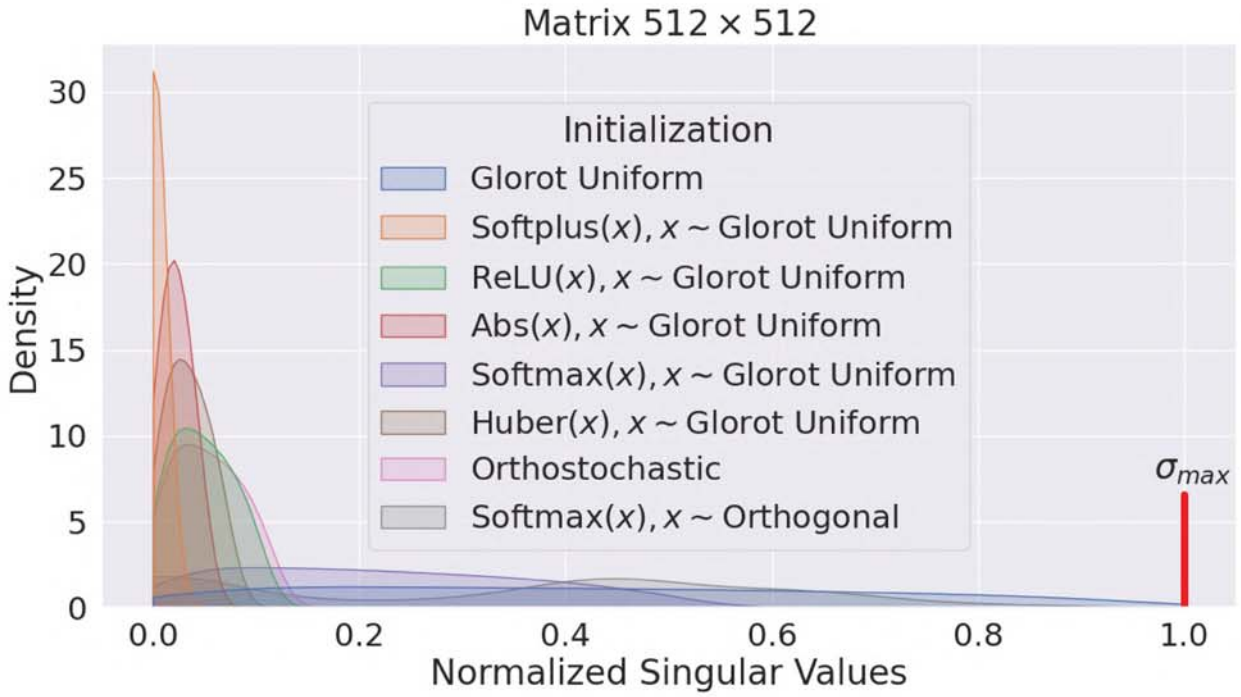


Figure 7.1: **Singular values of random matrices under various parametrizations of the non-negative orthant** $\mathbb{R}_+$. The singular values are normalized such that $\sigma_{\max} = 1$.

As a consequence, the differentiable function $T = \nabla_x f$ *fulfills:*

$$\left(\frac{\partial T}{\partial x}\right) = \left(\frac{\partial T}{\partial x}\right)^T. \tag{7.5}$$

Property 8 is the motivation behind the regularization term ϕ

$$\phi(T_\theta) := \mathbb{E}_{x\sim\mathcal{X}}[\| \left(\frac{\partial T_\theta}{\partial x}\right) - \left(\frac{\partial T_\theta}{\partial x}\right)^T \|_F], \tag{7.6}$$

typically used in OT (Uscidda and Cuturi, 2023), or for generalization guarantees (Cui et al., 2022). Unfortunately, like any regularization, it fails to give the formal guarantee that $T = \nabla_x f$ for some f, and it doesn't get rid of the need for nested differentiation $\nabla_\theta \phi(T_\theta)$ to optimize the parameters θ. Furtherore, it does not enforce the convexity constraint either. This leads to the question, of whether is it possible to parametrize gradient fields *without* nested autodifferentiation.

Warning 7.1. Stability of gradient field under composition.

Let $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^n$ be two gradient fields, i.e. $f = \nabla_x F$ and $g = \nabla_x G$ for smooth and differentiable scalar functions F and G. Then $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial x}^T$ and $\frac{\partial g}{\partial x} = \frac{\partial g}{\partial x}^T$. We define $h = f \circ g$. Then in general we have

$$\left(\frac{\partial h}{\partial x}\right) \neq \left(\frac{\partial h}{\partial x}\right)^T, \tag{7.7}$$

that is, gradient fields are not stable under composition. There is no $H : \mathbb{R}^n \to \mathbb{R}$ such that

$h = \nabla_x H$. This is a direct consequence of the fact that symmetric matrices are not stable under products.

For two symmetric matrices $A = A^T$ and $B = B^T$, their product AB is symmetric if and only if the matrices commute: $AB = BA$. Back to our original problem, this implies that the Jacobian $\frac{\partial f}{\partial x}$ and $\frac{\partial g}{\partial x}$ of the corresponding functions must commute.

Open question. What are the set of smooth functions $\mathcal{F}$ such that for all $f \in \mathcal{F}$:

- $\left(\frac{\partial f}{\partial x}\right) = \left(\frac{\partial f}{\partial x}\right)^T$, i.e. there exists F such that $f = \nabla_x F$.
- for all $g \in \mathcal{F}$, with $h := f \circ g$, we have $\left(\frac{\partial h}{\partial x}\right) = \left(\frac{\partial h}{\partial x}\right)^T$.

This boils down to finding a group of symmetric matrices G that commute. And then to find all the *smooth* functions $\mathcal{F}$ such that $\frac{\partial f}{\partial x} \in G$. The smoothness condition is the most severe. We perform a first attempt with *symmetric circulant matrices*.

Exemple 7.3. Symmetric circulant matrices.

Symmetric circulant matrices are matrices of the form (in the case of 5×5 matrices):

$$C = \begin{bmatrix} a & b & c & b & a \\ b & a & b & c & b \\ c & b & a & b & c \\ b & c & b & a & b \\ a & b & c & b & a \end{bmatrix}. \tag{7.8}$$

They are symmetric by construction. Circulant matrices are commutative. Therefore, symmetric circulant matrices form a (commutative) ring, that we note $\mathcal{G}$, under matrix multiplication (Chao, 1988; Rojo and Rojo, 2004). This manifold is of dimension $\lceil \frac{n}{2} \rceil$, which is extremely low compared to the ambient space of dimension $n \times n$.

Therefore, it is possible to parametrize linear layers as $\mathcal{F} := \{f(x) = Cx + b | C \in \mathcal{G}, b \in \mathbb{R}^n\}$. It remains to find smooth activation functions whose Jacobian exhibits such structure. Unfortunately, enforcing smoothness (and in particular, continuity) with **non-linear** functions having this structure is not trivial. And without smoothness, the condition of property 8 do not apply. One way to solve the issue would be to relax the problem and look for *piecewise affine* activations, instead of smooth ones. To this day, this question is still open.

We can mention the works of Araujo et al. (2019, 2018) that studied universal approximation of functions with circulant matrices, leveraging their computational efficiency, with applications to video classification. Notably, with ReLU activation, "diagonal circulant networks" of unbounded width are universal approximators. However, their work does not address the design of activation functions with circulant matrices. However, it gives some insights: the product of circulant matrices is typically not full rank, which may be an issue for deep networks. We can also mention the recent work of Richter-Powell et al. (2022) regarding the neural parametrization of divergence-free fields. Finally, we can mention the work of Chaudhari et al. (2023) that defines a neural network whose Jacobian is positive definite, which effectively ensures that the map is the gradient of a convex function. Unfortunately, the universal approximation theorem does not hold for their construction (or at least, not trivially), since the network only performs a single linear transformation of the input

before stacking non-linearities: its expressiveness is reduced.

7.2.2 Gradient of a convex function

Assume that $H : \mathbb{R}^d \to \mathbb{R}$ is a real function whose *gradient* is 1-Lipschitz. We let

$$F_H(x) := \frac{1}{2}\|x\|_2^2 - H(x). \tag{7.9}$$

We let $h(x) = \nabla_x H(x)$ and $f_h(x) = \nabla_x F_H(x)$. Then we have

$$f_h(x) = x - h(x) \tag{7.10}$$

with $h : \mathbb{R}^d \to \mathbb{R}^d$ a 1-Lipschitz function. Then, the function F_H is convex, since $\frac{\partial^2 F_H}{\partial x^2} = I - \frac{\partial h}{\partial x} \succcurlyeq \mathbf{0}$. Therefore, if it is possible to enforce h to be *both* 1-Lipschitz *and* a gradient, then it is possible to parametrize directly the gradient of a convex function[1]. Nevertheless, current parametrizations of 1-Lipschitz functions (as the ones exposed in Chapter 2) do not verify the condition $\frac{\partial h}{\partial x}^T = \frac{\partial h}{\partial x}$. This is arguably a problem harder than the one exposed in the previous section.

7.3 Computation of Monge maps for squared Euclidean cost

In this section, we illustrate the use of convex neural networks in the context of optimal transport. As mentioned before, convex networks arise naturally in the context of Wasserstein-2 transportation plan learning. First, we recall briefly the optimization problem studied by González-Sanz et al. (2022):

González-Sanz, A., De Lara, L., Béthune, L. and Loubes, J.M.. **GAN estimation of Lipschitz optimal transport maps**, *2022*.

This work relies on Lipschitz constrained generators (and discriminators) to learn optimal transport maps. My contribution was the implementation of the algorithm in Tensorflow. Then, we expose a modification of this algorithm that involves non-Lipschitz optimal transportation maps (unpublished). After, we showcase an application to fairness with counterfactuals. We conclude with an application to the *center-outward map* that can be used to generalize quantiles to the multivariate case.

7.3.1 Lipschitz Monge maps

Given two distributions P and Q living in $\mathbb{R}^d$, with compact support (for simplicity), González-Sanz et al. (2022) solve:

$$T^\lambda(P, Q) \in \underset{T:\mathbb{R}^d \to \mathbb{R}^d}{\arg\min}\ \lambda \mathcal{W}_1(T_\# P, Q) + \mathbb{E}_{x\sim P}[\|T(x) - x\|_2^2]. \tag{7.11}$$

We recall that the Monge problem for squared Euclidean cost is the solution to the following problem:

$$T^*(P, Q) \in \min_{T_\# P = Q} \mathbb{E}_{x\sim P}[\|T(x) - x\|_2^2]. \tag{7.12}$$

Therefore, the problem of Equation 7.11 is a regularized version of Equation 7.12. Since the two objectives are antagonists, this induces a bias in the solution $T^\lambda(P, Q) \neq T^*(P, Q)$. Fortunately, if $\lambda \to +\infty$ then $T^\lambda \to T^*$. The advantage of using neural networks for parametrizing T_θ is

[1] I would like to thank Edouard Pauwel for this suggestion, which opens interesting perspectives.

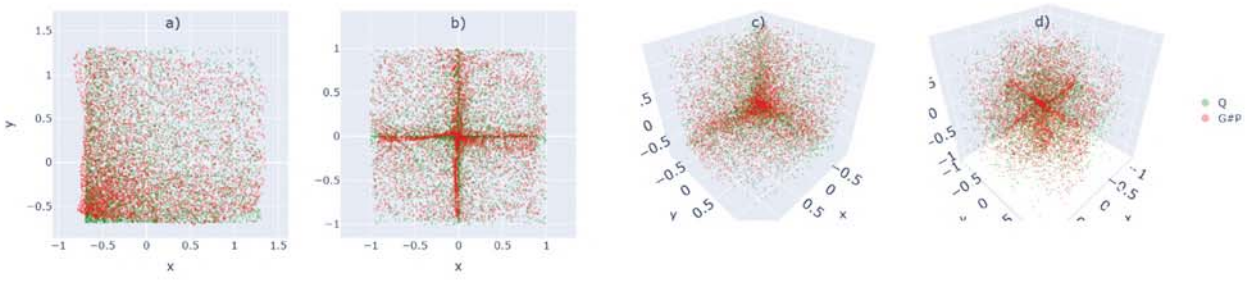


Figure 7.2: **Visualisation of $G_\sharp P$ and $Q := T_{0\sharp}P$ with $10,000$ points.** P is the uniform distribution on $[-1,1]^d$. The generator is trained for 120 gradient steps. The Figures (a)-(b) corresponds to $d = 2$. The Figures (c)-(d) corresponds to $d = 3$. In Figures (a)-(c), we defined T_0 by coordinate-wise application of $x \mapsto \frac{1}{1.18}(\exp x - 1.18)$. In Figures (b)-(d), we defined T_0 by coordinate-wise application of $x \mapsto x^2\text{sign}(x)$.

their ability to generalize outside the train set. Indeed, only empirical distributions $P_n \sim P^{\otimes n}$ and $Q_n \sim Q^{\otimes n}$ are available. In this case, for the empirical estimator $T^\lambda(P_n, Q_n)$ to converge to $T^*(P,Q)$, the regularization factor λ_n must grow to $+\infty$ at the "right" speed. Too slow, and the push forward constraint $T_\# P = Q$ might not be fulfilled. Too fast, and T might not minimize the squared Euclidean cost. This speed depends on the way the function T_θ is parametrized. In the work, González-Sanz et al. (2022) proposes to rely on a Lipschitz generator, since it gives statistical consistency results. See examples in Figure 7.2. However, this also induces strong constraints on the transportation plans that can be represented. Indeed, even for simple distributions P and Q, the Lipschitz constant of the Monge map T^* can be high (Salmona et al., 2022). Therefore, the algorithm of González-Sanz et al. (2022) can fail in situations where T is not Lipschitz, like in Figures 7.3. Uscidda and Cuturi (2023) solves the problem of tuning λ using a third term $-\mathcal{W}_2(P, T_\# P)$, that corrects the bias induced by λ. We can also mention the work of Fan et al. (2023) that studies a similar formulation with a dual Lagrange multiplier for the pushforward constraint. Nonetheless, the problem studied in González-Sanz et al. (2022) is interesting since it gives statistical consistency results involving the depth and the width of the network, which is uncommon in deep learning. In the next section, we study a setting in which we relax the Lipschitness constraint on T (which removes the statistical consistency results) to improve the empirical performance.

7.3.2 Non-Lipschitz Monge maps

In this section, we propose a different algorithm by relaxing the constraints on T. We do not enforce the Lipschitzness of T. All the preliminary results in this section and onward are unpublished personnal work.

Instead of using a Lipschitz generator T, we parametrize T as $T_\theta = \nabla_x f_\theta$, where f_θ is a input convex neural network, like defined in section 7.1. And since the samples P_n and Q_n are of finite size $n \in \mathbb{N}$, we consider λ as a hyper-parameter we optimize manually. Therefore we lose some statistical guarantees brought by the Lipschitz constraints on T. In exchange, T_θ can parametrize discontinuities since f_θ is now allowed to have discontinuous derivatives. For the estimation of $\mathcal{W}_1(T_\# P, Q)$ we fit a 1-Lipschitz neural network, and we optimize the Kantorovich-Rubinstein dual objective. We obtain a min-max formulation, typical of the GAN framework:

$$\underset{f \in \text{convex}(\mathbb{R}^d,\mathbb{R})}{\arg\min} \; \mathbb{E}_{x\sim P}[\|\nabla_x f(x) - x\|_2^2] + \lambda \max_{p \in \text{Lip}_1(\mathbb{R}^d,\mathbb{R})} \left(\mathbb{E}_{x\sim P}[p(\nabla_x f(x))] - \mathbb{E}_{x\sim Q}[p(x)]\right). \tag{7.13}$$

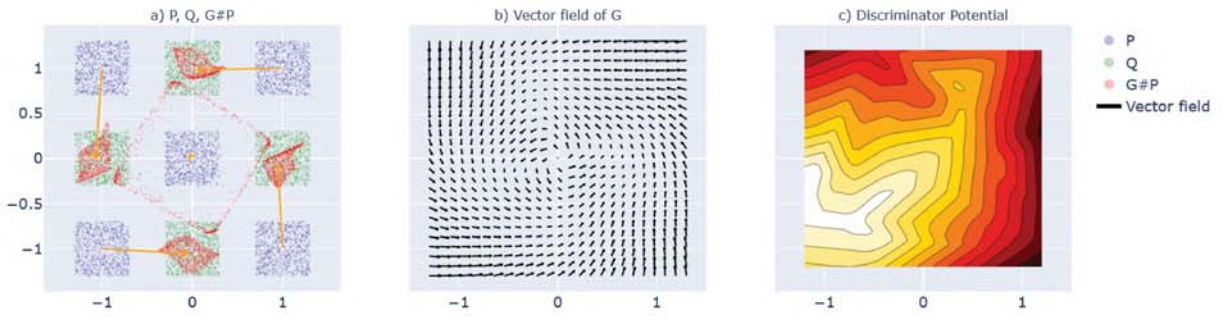


Figure 7.3: **Failure cases for squared Euclidean transportation cost**, with non-Lipschitz (even non-continuous) optimal Monge maps. We see that the Lipschitz generator G struggles to fulfill the push forward constraint $G\#P = Q$.

Like many min-max formulations, we rely on the amortized optimization framework Amos et al. (2023): we re-use the previous optimum p_t^* of the t-th step as a high-quality initialization for the $t+1$-th step. This makes the inner loop cheaper than solving a whole optimization problem from scratch. The procedure is sketched in Algorithm 10.

7.3.3 Counterfactual fairness

Among applications of optimal transport, fairness stands out. Indeed, if two groups of people, modelized by distributions P and Q, receive unfair treatment, for example in accessing a bank loan, predicting risks of recidivism, getting hired for a job, etc... it is possible to fix the issue by *transporting* individuals from the discriminated group to the normal group, using transportation map T. This pre-processing of data allows to transformation of an unfair classifier f into $f \circ T$. This is the "counterfactual fairness" solution democratized in Kusner et al. (2017) and Black et al. (2020). This is best explained with the words of De Lara et al. (2021):

> A *counterfactual* states how the world should be modified so that a given outcome occurs. For instance, the statement *had you been a woman, you would have gotten half your salary* is a counterfactual relating the *intervention* "had you been a woman" to the *outcome* "you would have gotten half your salary"

.

Exemple 7.4. Lipton synthetic dataset

We rely on the *Lipton synthetic dataset* (Lipton et al., 2018). In this synthetic dataset, each individual is characterized by three features W, H, Y with W the *work experience*, H the *hair length*, and $Y \in \{0, 1\}$ a boolean that indicates whether or not the individual was hired. Furthermore, the individuals are split into two groups: Male and Female. This is a hidden variable G that has effects on both work experience and hair length. This description is given in the seminal paper:

> This data-generating process has the following key properties: **(i)** the historical hiring process was based solely on the number of years of work experience; **(ii)**

Algorithm 10 GAN learning of Monge map

Input: source distribution P, target distribution Q, regularization parameter λ, Lipschitz discriminator $\{D_\psi\}_{\psi\in\Psi}$, input convex neural network f_ϕ, generator $\{T_\phi\}_{\phi\in\Phi} := \nabla_x f_\phi$, respective learning rates η_D and η_T, minibatch size m

repeat

repeat

Sample minibatchs: $\{x_i\}_{i=1}^m \sim P$, $\{y_i\}_{i=1}^m \sim Q$

Define cost function:

$$\mathcal{W}_D(\psi) := \frac{1}{m}\sum_{i=1}^m D_\psi(T_\phi(x_i)) - \frac{1}{m}\sum_{i=1}^m D_\psi(y_i)$$

Projected gradient ascent step on discriminator:

$$\psi \leftarrow \mathcal{P}_\Psi\left(\psi + \eta_D \nabla_\psi \mathcal{W}_D(\psi)\right)$$

until convergence of D_ψ

Sample minibatch: $\{x'_i\}_{i=1}^m \sim P$

Define cost functions:

$$\mathcal{W}_T(\phi) := \frac{1}{m}\sum_{i=1}^m D_\psi(T_\phi(x'_i))$$

$$\mathcal{C}(\phi) := \frac{1}{m}\sum_{i=1}^m \|x'_i - T_\phi(x'_i)\|^2$$

Projected gradient descent step on generator:

$$\phi \leftarrow \mathcal{P}_\Phi\left(\phi - \eta_T \nabla_\phi(\mathcal{C}(\phi) + \lambda \mathcal{W}_T(\phi))\right)$$

until convergence of G_ϕ

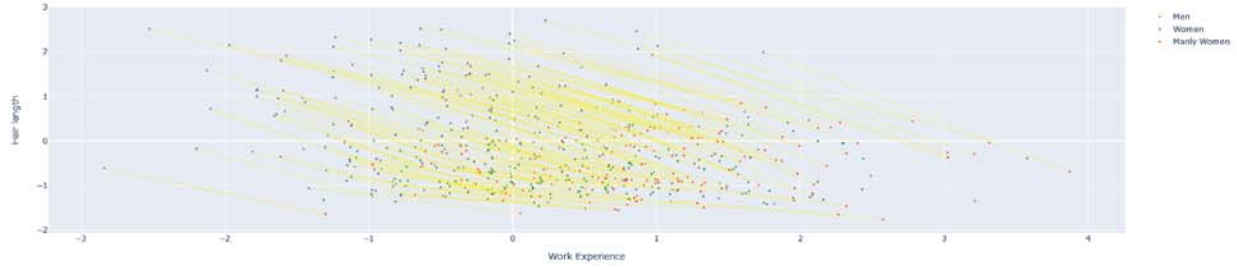
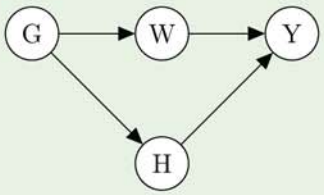


Figure 7.4: **Counterfactuals based on optimal transport** on Lipton dataset (Lipton et al., 2018), computed with Algorithm 10. The transportation T is highlighted in yellow. Counterfactuals are denoted "manly women" in the legend.

because women on average have fewer years of work experience than men (5 years vs. 11), men have been hired at a much higher rate than women; and **(iii)** women have longer hair than men, a fact that was irrelevant to historical hiring practice.

Since the hair length feature is correlated to the label y, and classifier may produce unfair decisions based solely on hair length rather than work experience. We illustrate this below:



Optimal transport can be used to produce counter-factual individuals, as illustrated in Figure 7.4 to solve the issue.

7.3.4 Multivariate Quantiles with Center-Outward distribution

We illustrate this algorithm on the task of multivariate quantiles estimation, with the so-called *center-outward* distribution. The literature around this object is plethora, see for example del Barrio et al. (2018); Figalli (2018); Beirlant et al. (2020); Hallin et al. (2020); del Barrio et al. (2022); Hallin et al. (2023). This tool allows to define a generalization of quantiles for the multivariate case $y \in \mathcal{Y} \subset \mathbb{R}^d$. The idea is the following:

1. Define the *spherical uniform* distribution U_d, defined as the product of the uniform over the unit sphere $\mathcal{S}_{d-1}$ with uniform over the unit interval of distances to the origin. The probability density function u_d of U_d verifies $u(r\theta) \propto r^{1/d}$, with $r \in [0,1]$ and $\theta \in \mathbb{R}^d$ with $\|\theta\|_2 = 1$. We note $\mathbb{S}_d$ the support of U_d.
2. Compute the Monge map T between U_d and a target distribution P by minimizing squared Euclidean cost. T is the *center-outward quantile map.*
3. We define the *quantile region of order* τ as:

$$\mathbb{C}_{\pm}(\tau) := T(\tau \mathbb{S}_d) \qquad \tau \in (0,1), \tag{7.14}$$

having the central property that:

$$\mathbb{P}(Y \in \mathbb{C}_{\pm}(\tau)) = \tau, \tag{7.15}$$

justifying the interpretation as a quantile region. Note that $\mathbb{C}_{\pm}(0)$ can be interpreted as a generalized median. Q *contours* and quantiles *tubes* can be defined similarly. See Hallin (2022) and references therein for more details.

Exemple 7.5. 2D multi-modal distributions.

We illustrate the procedure on 2D mixtures of Gaussians, taking inspiration from the examples of Hallin et al. (2021). We consider three tasks:

1. **Mixture** $\frac{3}{8}\mathcal{N}(\mu_0, \Sigma_1) + \frac{3}{8}\mathcal{N}(\mu_0, \Sigma_2) + \frac{1}{4}\mathcal{N}(\mu_0, \Sigma_3)$
2. **Mixture** $\frac{3}{8}\mathcal{N}(-3\mu_h, \Sigma_1) + \frac{3}{8}\mathcal{N}(3\mu_h, \Sigma_2) + \frac{1}{4}\mathcal{N}(-\frac{5}{2}\mu_v, \Sigma_3)$
3. **Mixture** $\frac{3}{8}\mathcal{N}(-8\mu_h, \Sigma_1) + \frac{3}{8}\mathcal{N}(8\mu_h, \Sigma_2) + \frac{1}{4}\mathcal{N}(-5\mu_v, \Sigma_3)$

with:

$$\mu_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \mu_h = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mu_v = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$\Sigma_1 = \begin{bmatrix} 5 & -4 \\ -4 & 5 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix} \quad \Sigma_3 = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}.$$

The report the empirical contours from the empirical center-map T_n computed with Algorithm 10, with $n = 4,000$. The value of λ is optimized with a grid search. Results are reported in figure 7.5.

7.4 Parametric Multivariate Quantile Regression

In this chapter, we discuss the possibility of performing multivariate quantile regression with statistical guarantees, using Lipschitz neural networks and convex neural networks. More precisely, this procedure takes inspiration from nonparametric multiple-output center-outward Quantile Regression from del Barrio et al. (2022), conditional Monge maps from Bunne et al. (2022), from the Monge gap regularizer Uscidda and Cuturi (2023), and 1-Lipschitz neural networks.

7.4.1 The meta-algorithm

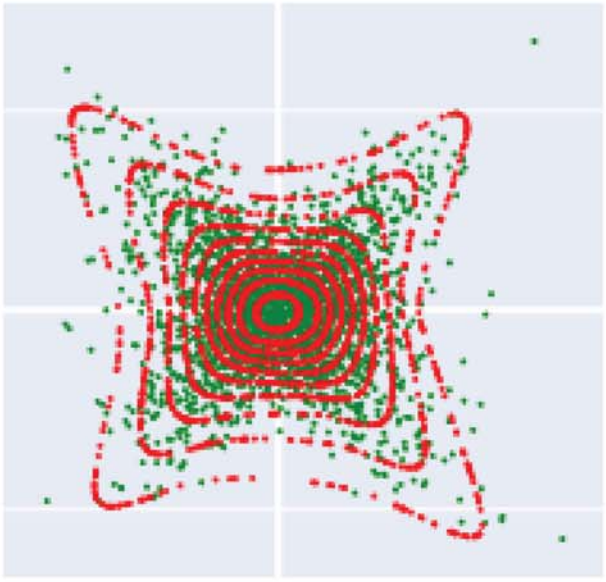
Given a dataset of n observations (x_i, y_i) with $x_i \in \mathcal{X} \subset \mathbb{R}^f$ and $y_i \in \mathbb{R}^d$, with $d > 1$, the goal is to estimate the quantile region of order τ conditionned by x, defined as:

$$\mathbb{C}_{\pm}(\tau | X = x) := T_x(\tau \mathbb{S}_d) \qquad \tau \in (0, 1), \tag{7.16}$$

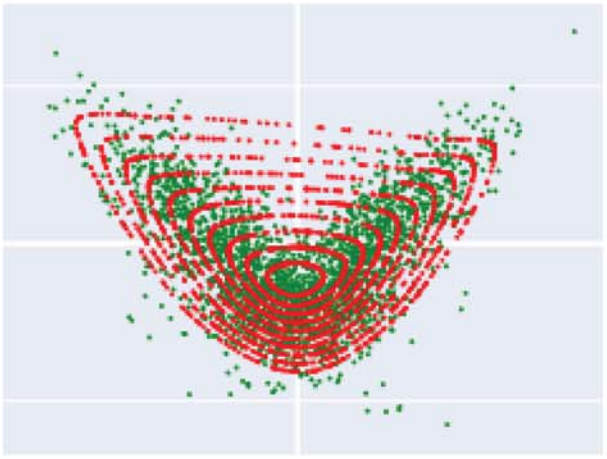
with the property that:

$$\mathbb{P}(Y \in \mathbb{C}_{\pm}(\tau | X = x)) = \tau. \tag{7.17}$$

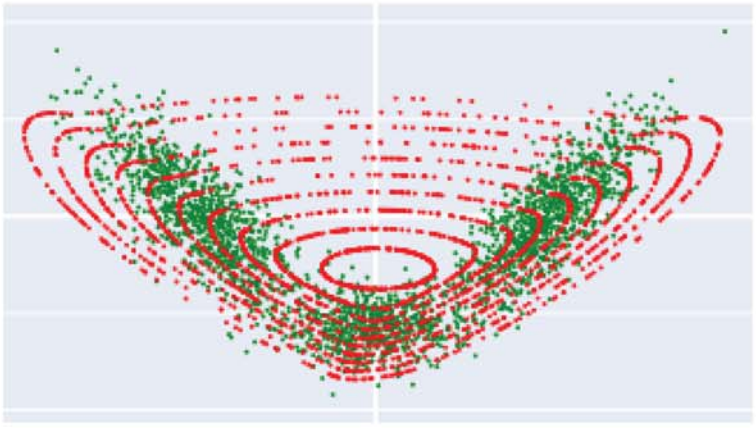
Refer to del Barrio et al. (2022) and references therein for more details on this construction. We see that this amounts to estimating a collection of transductive optimal transport maps T_x for each $x \in \mathcal{X}$. Conditional Monge maps have been studied by Bunne et al. (2022) and the recent work



(a) **Mixture** $\frac{3}{8}\mathcal{N}(\mu_0, \Sigma_1)+\frac{3}{8}\mathcal{N}(\mu_0, \Sigma_2)+\frac{1}{4}\mathcal{N}(\mu_0, \Sigma_3)$.



(b) **Mixture** $\frac{3}{8}\mathcal{N}(-3\mu_h, \Sigma_1) + \frac{3}{8}\mathcal{N}(3\mu_h, \Sigma_2) + \frac{1}{4}\mathcal{N}(-\frac{5}{2}\mu_v, \Sigma_3)$.



(c) **Mixture** $\frac{3}{8}\mathcal{N}(-8\mu_h, \Sigma_1) + \frac{3}{8}\mathcal{N}(8\mu_h, \Sigma_2) + \frac{1}{4}\mathcal{N}(-5\mu_v, \Sigma_3)$

Figure 7.5: **Empirical quantile contours** obtained by computing the Monge map that minimize squared Euclidean cost, between the spherical uniform distribution U_d and a mixture of Gaussians defined by $\mu_0 = \binom{0}{0}, \mu_h = \binom{1}{0}, \mu_h = \binom{0}{1}, \Sigma_1 = \begin{pmatrix} 5 & -4 \\ -4 & 5 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 5 & 4 \\ 4 & 5 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}$.

of Manupriya et al. (2023). Neural networks offer a natural way to parametrize those conditional transportation maps.

Definition 20 (Neural Conditional Monge map)**.** *We define the conditional Monge map as a function:*

$$\mathcal{T} : \mathcal{X} \times \mathcal{Y} \times \Theta \to \mathcal{Y} \tag{7.18}$$

where $\mathcal{X}$ is the input space, $\mathcal{Y}$ the prediction space, and Θ the parameter space. The function will be typically denoted as $\mathcal{T}_\theta(\cdot|x)$, or $\mathcal{T}(\cdot|x)$ when the parameters θ are obvious from context.

The map $\mathcal{T}$ solves the conditional OT problems $(\mathcal{T}_x)_\# U_d = \mathbb{P}(Y|X = x)$, and in particular regresses the "generalized median" $\mathbb{C}_\pm(0|X = x)$.

Exemple 7.6. Parametrization of Neural Conditional Monge maps.

Different strategies are possible to parametrize $\mathcal{T}$. We detail below some examples:

1. **Multiple inputs strategy**: T_θ is the neural network operating on the concatenated input $[x, y]$.
2. **Multiple inputs strategy**: $T_\theta(y|x) = \nabla_y E_\theta(y|x)$ is the gradient of a Partially Input Convex Neural Network (PICNN) as defined in Amos et al. (2017).
3. **Meta-learning strategy**: $T_\theta(y|x) = g_{f_\theta(x)}(y)$ where $f_\theta(x)$ are the parameters of the network $g(\cdot)$, and f_θ is itself a neural network.

These parametrizations leverage different implicit biases and offer different opportunities for generalization or constraint enforcement.

On most datasets, because of measurement errors and scarcity of data, each x_i is unique. In this setting, each of the individual OT problem $(\mathcal{T}_{x_i})_\# U_d = \mathbb{P}(Y|X = x_i)$ is trivial since $\mathbb{P}(Y|X = x_i)$ is the Dirac δ_{y_i}. However, this strategy prevents meaningful generalization outside of the train set: this cannot be a reliable way to build a *statistically consistent* estimator. This is where the parametrization of $\mathcal{T}$ comes into play: the constraints on $\mathcal{T}$ must be chosen such that fitting the Diracs is impossible. For example, it is possible:

1. to enforce K-Lipschitz constraint on $x \mapsto T(\cdot|x)$,
2. to enforce $y \mapsto T(y|x)$ to be invertible and to be partially *measure preserving*, taking inspiration from the *normalizing flows* literature (Kobyzev et al., 2020) by regularizing the log-determinant $\log\det(\frac{\partial \mathcal{T}}{\partial y})$.

The meta-algorithm is given in Algorithm 11. All the methods discussed in Korotin et al. (2021), Korotin et al. (2022) or Uscidda and Cuturi (2023) are candidate solutions to solve each of the conditional OT problems. They often involve complicated min-max formulations or nested optimization problems. A benchmark is necessary to settle the question. The ingredients necessary for this algorithm are summarized in Figure 7.6.

Remark 7.3. Instanciation of the meta-algorithm in a practical algorithm.

It is critical for these methods to be sample-efficient, considering the particular nature of the problem. A few comments can be made on this algorithm 11:

- the parametrization of $\mathcal{T}$ plays an important role in the implicit bias,
- the loss $\mathcal{L}$ depends on the algorithm chosen. It involves only two samples: one from $\mathbb{P}(Y|X = x_i)$ and one from the spherical uniform U_d.

Item 2 is to ensure that the runtime costs increase linearly with the batch size and the dataset size. This is to remain compliant with the paradigm of deep learning (discussed in chapter 1)

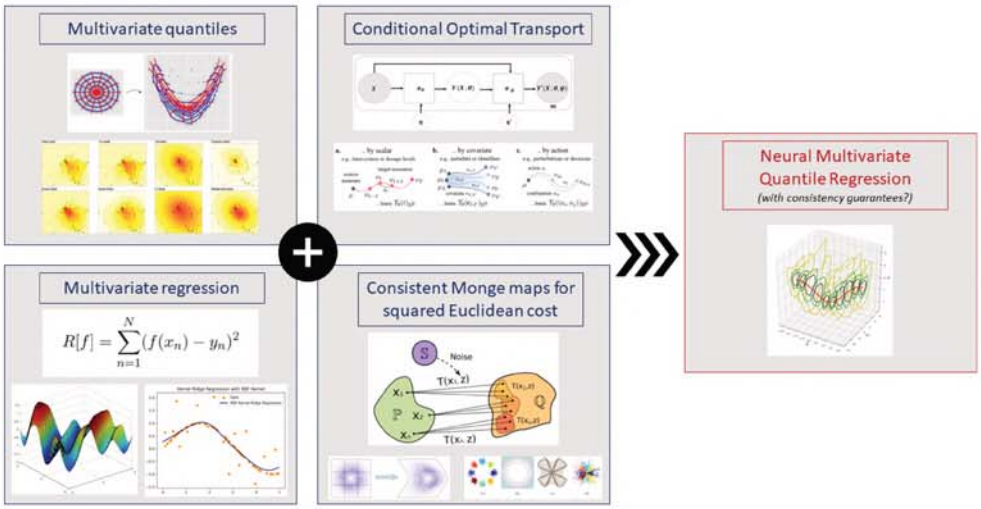


Figure 7.6: **Illustration of the proposed "meta-algorithm" for neural multivariate quantile regression**. Embedded graphics come from Brilhault et al. (2023); Bercu et al. (2023); Bunne et al. (2022); Manupriya et al. (2023); Evans and Hughes (2013); del Barrio et al. (2022). The resulting sketched in Algorithm 11 leverages tools from different fields.

and ensure that the algorithm can be applied to enormous datasets. Identifying the best $\mathcal{L}$ and $\mathcal{T}$ is a research project on its own, currently ongoing.

Algorithm 11 Multivariate Quantile Regression with Neural Conditional Monge maps

Input: Neural Conditional Monge maps $\mathcal{T}_\theta(\cdot|\cdot)$
Input: Dataset $\mathcal{D} = (x_i, y_i)$
Input: Loss $\mathcal{L} : \Theta \times \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, learning rate η

1: **for** each mini-batch $\mathcal{B}$ from $\mathcal{D}$ **do**
2: Sample a mini-batch $u \sim U_d^{\otimes b}$.
3:
4: Perform a gradient step $\theta_{t+1} \leftarrow \theta_t - \eta \nabla_\theta \frac{1}{|\mathcal{B}|} \sum_{i=1}^{b} \mathcal{L}(\mathcal{T}_\theta, x_i, y_i, u_i)$.
5: **end for**
6: **return** $\mathcal{T}_\theta(\cdot|\cdot)$

7.5 Perspectives

Here, we discuss the links of multivariate quantile regression with Conformal prediction (CP), and the possible future works.

7.5.1 Links with Conformal Prediction

The field of conformal prediction aspires to provide statistical guarantees in various prediction tasks with minimal hypotheses on the underlying algorithm. Vanilla conformal prediction typically requires a lot of training rounds, or leave-one out estimators like *Jacknife* and its variants (see Fontana et al.

(2020) and references therein). In practice, a *calibration set* (in the spirit of temperature calibration studied in section 4.4) is often used to estimate the confidence intervals. We see that our approach is different: the uncertainty is at *train time*, and this is the "intrinsic" uncertainty of the task (i.e. the noise ϵ in the observations $Y := \hat{f}(X) + \epsilon$ where $\hat{f}(X)$ is the ground truth, and $\mathbb{E}[\epsilon|X] = 0$ the noise of the observations). The intervals given by our algorithm are sensitive to some hyper-parameters like the Lipschitz constant K of the map $x \mapsto T_x(z)$ and don't provide guarantees. Nonetheless, the two approaches can be combined, and maybe it is possible to leverage some properties of the Center Outward map to improve the confidence intervals, in the same spirit of Conformal Gaussian Process Regression (Papadopoulos, 2023). This is an example of future work.

7.5.2 Conclusion

This chapter presented openings and future works on the topic of constraints in deep learning, and their link with optimal transport. More specifically it focused on convexity constraints, and their link with optimal transport with **quadratic** Euclidean cost. We showed that optimizing convex input neural networks was harder than conventional networks due to the degenerated spectrum of the weight matrices. Moreover, we discussed the possibility of a *direct* parametrization of gradients, without relying on

Chapter 8

Conclusion

In this last chapter, we take a step back on some of the topics explored in the thesis, and we attempt to replace them in a broader context. We hope that these high-level remarks can serve as a basis for future research directions.

1. **First**, we discuss the ever-lasting debate of *constraints* versus *regularization*, in the light of Lipschitz networks.
2. **Then**, taking inspiration from chapter 6, we perform a short literature review of some algorithms inspired by back-propagation. We also discuss what could be the future extension of this framework.
3. **After**, we discuss an understudied framework for generalization: the Kolmogorov complexity and some variants (Minimum Description length, ν-information...). We show how it may address some shortcomings of existing frameworks that theorize generalization.
4. **Finally**, we attempt to *define* AGI[1] such as commonly understood in the media, and what are the current obstacles to its existence.

These sections, and the chapter 7 as a whole, are also intended as draft of *research statement* for future works. The tone is less rigorous and precise than the rest of the manuscript on purpose, as excessive rigor may negate creativity on ill-defined questions.

Contents

[1] AGI: Artificial General Intelligence.

8.1 Which future for Lipschitz networks?

In this section, we take a step back and discuss the limitations of Lipschitz networks. We identify here the main obstacles to their success.

8.1.1 Bias of the function space

In the context of classification, chapter 4 largely addresses this and concludes that this function space is not limited. However, for other tasks like regression, representation learning, object detection, and segmentation, this is less obvious. Segmentation is akin to a multi-label classification task, with as many labels as input pixels. Object detection is similar to segmentation, with the additional difficulty of regressing the position and dimensions of boxes. Regression tasks are the most challenging: the Mean Square Error can only be minimized if the target function has the same Lipschitz constant as the Lipschitz network. When the observations $y = f(x) + \epsilon$ are noisy with $\mathbb{E}[\epsilon|x] = 0$, priors on the true Lipschitz constant $L(f)$ can be used as a regularization to the regressed function $\hat{f}$. This shares a similar spirit to Gaussian processes, for which the prior takes the form of a kernel function $k(\cdot, \cdot)$. For all these topics, it is not clear if the Lipschitz constraints are beneficial. As the training progresses the Lipschitz constant increases (see Chapter 4), which in practice is not only the augmentation of the leading singular value $\sigma_{\max}$, but the whole singular values spectrum σ_i. This suggests that constraints based on singular values clipping like in Ebrahimpour-Boroojeny et al. (2023) can mitigate the phenomenon: the mapping $\sigma_i \mapsto \sigma_i/\sigma_{\max}C$ is replaced with $\sigma_i \mapsto \min(\sigma_i, C)$, where C is the maximum singular value.

Arguably, the hardest theoretical task is not quantifying the expressiveness of the function space (since Lipschitz functions in general are widely used and studied in the learning theory), but characterizing the implicit bias induced by a given parametrization. Below, we discuss a popular tool used to characterize the implicit bias of SGD in neural network training: the *Neural Tangent Kernel*.

8.1.2 Is there a Lipschitz neural tangent kernel?

The concept of "Neural Tangent Kernel" (NTK) has been introduced by Jacot et al. (2018). It was intended as a tool to understand the implicit bias of SGD in deep (and more importantly, *wide*) neural networks. More precisely, this work connects the neural networks training (which is highly non-convex) with kernel methods, and more specifically kernel regression, that are convex. They study the training into *function space* f_t rather than parameter space θ_t. It happens that the cost is convex in function space, which facilitates the analysis. The work stems from the observation that at initialization, a random neural network defines a valid (positive definite) kernel. This was already known in previous work like Neal (2012) or Lee et al. (2018a).

Gaussian Processes with Lipschitz constraints. The first remark of outermost importance is that there are no Gaussian Processes $\mathcal{P}(m, k)$ and no value $K > 0$ such that for all $f \sim \mathcal{GP}(m, k)$ the function f is K-Lipschitz. The GP prior enforces some smoothness with high probability, but in general, the posteriors of the GP can exhibit arbitrarily high Lipschitz constant[2]. Therefore, it seems unlikely that kernel methods could marry well with Lipschitz constraints.

[2] I would like to thank François Bachoc for these insights.

Initialization in the Neural Tangent Kernel. The whole NTK construction relies on the hypothesis that the network defines a well-behaved kernel *at initialization*. The importance of initialization in neural networks is often overlooked[3]. Crucially, the NTK framework assumes the following definition of neural network:

$$\begin{aligned} h_0(x) &:= x, & z_{l+1}(x) &:= \frac{1}{\sqrt{n_l}} W_l h_l(x) + \beta b_l, \\ h_l(x) &:= \sigma(z_l(x)), & f([W_0, W_1, \ldots], x) &:= z_L(x). \end{aligned} \tag{8.1}$$

where $\beta > 0$ is some hyper-parameter and $W_t \in \mathbb{R}^{n_{l+1} \times n_l}$. Here, $\frac{1}{\sqrt{n_l}}$ is a rescaling factor. It is introduced to define the behavior of W in the infinite width limit $n_l \to +\infty$. Finally, it is assumed that $W \sim \mathcal{N}(0, 1)$, i.e. the entries are random iid Gaussians. The singular values of W follow the Marchenko-Pastur distribution (Marchenko and Pastur, 1967). And according to Rudelson and Vershynin (2010), the maximum singular value of W_l will grow as

$$\mathbb{E}[\sigma_{\max}(W_l)] \sim \sqrt{n_{l+1}} + \sqrt{n_l}. \tag{8.2}$$

Therefore we have

$$\mathbb{E}[\sigma_{\max}(\frac{1}{\sqrt{n_l}} W)] \sim 1 + \sqrt{\frac{n_{l+1}}{n_l}}. \tag{8.3}$$

However, these results hold with probability 1 only asymptotically, by taking the limits $n_1, n_2, \ldots n_D \to +\infty$ sequentially. Remember that n_0 describes the dimension of the input data, and therefore, n_0 remains fixed throughout the training. We see that the Lipschitz constant of $\frac{1}{\sqrt{n_0}} W_0$ growths as $\Theta(\sqrt{n_1})$. Because the limits $n_1, n_2, \ldots, n_D \to +\infty$ are taken sequentially, it suggests that further layers will suffer from the issue too. It may be tempting to fix the issue by replacing the factor $\frac{1}{\sqrt{n_l}}$ with $\frac{1}{\sqrt{n_l}+\sqrt{n_{l+1}}}$. This resembles the Glorot initialization, which uses the rule $\frac{\sqrt{6}}{\sqrt{n_l+n_{l+1}}}$ instead. It is worth noticing that:

$$|\sqrt{n_{l+1}} - \sqrt{n_l}| \leq \sqrt{n_{l+1} + n_l} \leq \sqrt{n_{l+1}} + \sqrt{n_l}, \tag{8.4}$$

and that $\mathbb{E}[\sigma_{\min}(W_l)] \sim |\sqrt{n_{l+1}} - \sqrt{n_l}|$ (Rudelson and Vershynin, 2010). From a high-level overview, it appears that the factor $\sqrt{n_{l+1} + n_l}$ corresponds more to an "average" behavior of $\frac{\|W_l x\|_2}{\|x\|}$ under $W_l \sim \mathcal{N}(0, 1)$, whereas Lipschitz certification is interested in the worst case behavior $\sqrt{n_{l+1}} + \sqrt{n_l}$.

Warning 8.1. Worst case vs average case?

From an epistemic point of view, certifying the worst-case behavior is often significantly harder than certifying the average behavior. This is notably the case in complexity theory, where it is easy to exhibit algorithms with low expected runtime (e.g. quick sort with a random pivot), but harder to come up with an algorithm that behaves correctly *in the worst case* (quick-sort with efficient computation of the median). This is also the case in robustness certification: randomized smoothing (Cohen et al., 2019) creates a *deterministic* classifier $\hat{f}(x) := \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}[f(x + \epsilon\delta)]$ with a larger certifiable robustness radius than using the Lipschitz constant of f alone (Delattre et al., 2023a), but when it is evaluated from a finite sample we obtain a random prediction $\hat{f}_n(x) := \frac{1}{n} \sum_{i=1}^{n} [f(x + \epsilon_i \delta)]$ and there is a non-zero probability to produce a wrong certificate.

[3] We can probably blame Glorot and Bengio (2010) and LeCun et al. (2002) for this, which did a too good of a job. As a result, most practitioners today are unaware of the importance of the initialization on the training.

If we apply the factor $\frac{1}{\sqrt{n_l}+\sqrt{n_{l+1}}}$ and if we apply the inductive reasoning in Proposition 1 of Jacot et al. (2018), by taking the sequential limit $n_1, n_2, \ldots, n_D \to +\infty$, we obtain that the output of the first layer is a centered Gaussian process with covariance

$$\Sigma^{(1)}(x, x') = \frac{1}{n_0 + n_1} x^T x' + \beta^2. \tag{8.5}$$

When $n_1 \to +\infty$ this covariance matrix collapses to zero, which is a degenerated GP. By induction, further layers are Gaussian process with zero mean and covariance matrix *independant* of the input. This is, once again, an example of "vanishing gradient" phenomenon. We see that, this phenomenon has little to do with gradients themselves (which eliminates the relevance of using tricks during back-propagation, like upscaling). The vanishing gradient phenomenon is more a consequence of the output being independent of the input, that is more an issue of the network during the *forward* pass, than an issue of the network during the backward pass.

Interpolating phenomenon. Finally, one of the motivations behind the NTK was to study the "interpolating" regime, in which the over-parametrized network $\hat{f}$ fits exactly each observation $\hat{f}(x_i) \approx f(x_i) + \epsilon_i$, and *generalizes well anyway*. The interpolating regime arises as a hypothesis in some stochastic optimization methods like SGD with Armijo line search (Vaswani et al., 2019). It is qualitatively different from the classical learning theory, in which a bias is introduced to reduce the variance of the estimator, which leads to non-zero residuals $\|y_i - \hat{f}(x_i)\|$. For regression tasks, Lipschitz leans more toward the classical regime than the interpolating one. In this sense, they can be considered closer to the classical learning theory than the theory of deep learning.

This does not preclude completely the possibility of applying NTK theory to Lipschitz-constrained networks, but it shows that significant obstacles must be overcome.

8.1.3 Reproducing Kernel Banach Spaces of Lipschitz functions

Kernels are a powerful tool, and but also arguably a very constrained one, because they correspond to an inner product in some infinite-dimensional function space. This blessing brings the representer theorem (Schölkopf et al., 2001a) and its variants, numerous generalization guarantees, and makes the learning procedure amenable to convex optimization. This blessing has a cost: not all similarity measures can be converted to a valid kernel. Similarly, not all complexity measures correspond to a Hilbert norm. In this section, we explore another direction: we explore Lipschitz through the lens of Reproducible Banach space (RKBS).

In an RKBS, instead of considering a Hilbert space with an inner product, we consider a Banach space $(\mathcal{B}, \|\cdot\|_{\mathcal{B}})$ with a non-Hilbertian norm. Here, we chose the Lipschitz constant as the complexity measure:

$$\forall f \in \mathcal{B}, \quad \|f\|_{\mathcal{B}} = \mathrm{Lip}(f). \tag{8.6}$$

We assume that $f : \mathcal{X} \to \mathbb{R}$ with $\mathcal{X} \subset \mathbb{R}^d$ being a compact domain (for simplicity). Note that this does not define a valid norm (yet): all constant functions have null Lipschitz constant, violating the positive definitiveness property. The solution is easy: it is sufficient to choose an arbitrary anchor point $x_0 \in \mathcal{B}$ and enforce $f(x_0) = \mathbf{0}$ for all $f \in \mathcal{B}$. This defined a "pointed" space, with x_0 the distinguished point. This strategy has been used in the seminal work of von Luxburg and Bousquet (2004).

Now, we can attempt to apply the theory of Reproducing Kernel Banach Space (RKBS) and see what sticks. In this regard the work of Micchelli and Pontil (2004), Zhang et al. (2009), and Lin et al. (2022) are particularly relevant. Another obstacle on the way is that defining a norm is not sufficient: "A normed vector space $\mathcal{B}$ is called a Banach space of functions on Ω if it is a Banach space whose

elements are functions on Ω, and for each $f \in \Omega$, $\|f\|_{\mathcal{B}}$ vanishes if and only if f, as function, vanishes everywhere on Ω" (Zhang et al., 2009). These additional properties need to be checked carefully.

Is it possible to create a Hilbert space? That seems unlikely. One of the issues is that the Lipschitz constant is fundamentally a *global* property, a supremum over the domain of the function. We saw that it could be used to define a norm, but alas, not all norms correspond to an inner product.

Remark 8.1. Banach or hidden Hilbert?

The parallelogram identity is a necessary and sufficient condition for a norm to be Hilbertian:

$$\|f+g\|_{\mathcal{B}}^2 + \|f-g\|_{\mathcal{B}}^2 = 2(\|f\|_{\mathcal{B}}^2 + \|g\|_{\mathcal{B}}^2). \tag{8.7}$$

In this case the inner product takes the form

$$\langle f, g\rangle_{\mathcal{H}} := \frac{1}{2}(\|f+g\|_{\mathcal{B}}^2 - \|f\|_{\mathcal{B}}^2 - \|g\|_{\mathcal{B}}^2). \tag{8.8}$$

However, it is possible to create a Hilbert space with an inner product using a measure related to the Lipschitz constant.

Exemple 8.1. Dirichlet energy.

For example, $\langle f, g\rangle_{\mathcal{H}} = \int_\Omega \langle \nabla_x f(x), \nabla_x g(x)\rangle dx$ is a good candidate. The associated norm $\|f\|\mathcal{H} = \int_\Omega \|\nabla_x f(x)\|_2^2 dx$ is the **Dirichlet Energy**, that was recently studied in the deep learning context (Dherin et al., 2022). Care must be taken of functions $f \neq g$ for which $\|f-g\|\mathcal{H} = 0$, i.e. functions that are different but have the same gradient almost everywhere w.r.t the Lebesgue measure. Quotienting this space (i.e. taking a "representer" for each class of equivalence) might be necessary to ensure it defines a valid RKHS.
In general, every positive definite bilinear form can be used as a basis to create a RKHS, from quantities related to the original function (e.g. the function itself, its gradient, or higher order derivatives), providing that the right quotient space is chosen.

8.2 Back-propagation and its variants

The root of the algorithm presented in chapter 6 is a variant of back-propagation, that backpropagates *bounds* instead of cotangent vectors. In general, back-propagation is an algorithm that traverses a *computation graph* to perform computations. This general principle can be applied to other topics. For example, certification methods based on abstract interpretation or bounding boxes (see Xu et al. (2020) and Fazlyab et al. (2019)) act on the forward computation graph. Derivatives can be computed in forward or backward mode. In literature, Klaus et al. (2022) compute *convexity certificates* from the Hessian. In Srinivasan and Todorov (2015) and Roulet et al. (2022) the computation graph is traversed to extract higher-order derivatives in order-2 optimization methods. In Roulet and Harchaoui (2019), the framework is used to compute the local smoothness of the functional cost by operating on the computation graph. Finally, in Roulet and Harchaoui (2021) the framework is used to make a non-smooth computation graph amenable to smooth optimization techniques.

Remark 8.2. How much do we owe to software?

The content of this section is subjective on purpose and intended as a discussion.
The success of modern deep learning can largely be attributed to the back-propagation algorithm in its implementation in various libraries like Autograd, Caffee, Theano, Pytorch, Tensorflow, and Jax. Similarly, the impact of scikit-learn project on the whole machine learning community cannot be overstated[a]. The flexibility of these frameworks fostered adaptability, exploration, and discoveries. The tools available to the researchers are largely responsible for the productive output of these researchers. **Firstly**, because pursuing an academic career requires a careful risk and time management policy, which biases research toward incremental contributions based on existing tools, and further away from exploratory projects that involve huge software engineering as a first step. **Secondly**, mastering a complex tool that has required years of development poses a challenge if the tool is not available as a package, as months of work may be required. This "entrance ticket price" can hinder cross-field contributions (for example, machine learning and health).
Since research contribution will primarily rely on readily available tools and much less on the ones that do not exist yet, it is absolutely essential for the community to produce new tools and abstractions and avoid being stuck in a "local optima" induced by obsolete packages. Creating and maintaining a framework is a significant amount of work that often goes beyond the individual contribution of a single researcher, which implies it must be either the responsibility of an institution, of a private company, or a shared responsibility through OpenSource. This also explains why the availability of the source code in the publication is a matter that goes beyond the reproducibility crisis: in computer science, the code *is* a contribution at least as much as the research itself[b].

[a] To echo the words of the recent public praise of scikit-learn by François Chollet (creator of Keras).

[b] In this regard, the DEEL project (Delseny et al., 2021) that hosts and maintains the libraries DeelLip (Lipschitz networks), Xplique (Explainability tools for deep learning), OODEEL (Out Of Distribution detection), puncc (conformal prediction), influenciae (influence functions for deep learning) is part of this process.

Exemple 8.2. The new guy in town: Jax.

In this regard, Jax appears as a promising framework that puts emphasis on the explicit control of Autodiff in forward *and* backward mode. Some of these innovations like *jacrev*, *jacfwd*, or *vmap* were brought to Pytorch 2.0. This increased flexibility helped recent developments like the "automatic implicit differentiation" of Jaxopt library. We can also mention Neural ODE (Kidger, 2021), differentiable physical engines (Freeman et al., 2021), or differentiable rendering , which were made possible (or at least facilitated) by Autodiff.

In light of these observations, it appears that the community must pursue the efforts in this direction. In the long term, if researchers want to take advantage of certification methods, or higher-order optimization methods, then it might be worth creating a new "generalized" Autodiff framework, whose purpose would not be confined to the computation of derivatives, but that could be transparently extended to other objectives. Below, is a sketch of what could be the API:

```
import computation_graph as cg
import computation_graph.numpy as cnp
import computation_graph.certify as cert
```

```
def f(x, y):
  xy = cnp.stack([x, y])
  c = cnp.array([1, 2])
  return cg.nn.relu(cnp.dot(xy, c)) - 2

# Examples of usage:
f.local_l2_lipschitz_constant(x=cert.l2ball(x0, radius=2.),
                              y=cert.l0ball(y0, radius=0.5))

# convexity certificates from hessians:
f.is_convex_on(x0=whole_domain(), y0=21.)

# classical back-propagation:
_, f_grad = f.vjp(x0, y0)
dloss_dx = f_grad(dloss)

# graph transformation: ReLU -> softplus
f_smooth = f.smooth_relaxation(smoothness_factor=0.75)

# certificates of smoothness
f_smooth.is_differentiable_on(x0=whole_domain(),
                              y0=whole_domain())  # yes!
```

Of course, this is a work that leverages the findings of several research projects for every method to be speed or memory-efficient. Every elementary operation (like add, sub, matrix-vector product, concatenate, etc...) could exhibit an interface with the following structure. This project would combine research and engineering practices. It could have the potential to be the basic tool in every toolbox for robust deep learning with formal guarantees - a bit like languages suited for formal verification in *old fashion*[4] software like Haskell (Vazou, 2016), Ocaml (Charguéraud et al., 2019) or Coq (Chlipala, 2022). The applications of this framework have been foreseen in previous chapters: robustness certification, training with privacy guarantees, or optimization with guarantees.

8.3 Selected pieces in learning theory

In this chapter, we perform a brief review of generalization theorems that exist in literature and perform a first attempt at classifying them. We place ourselves in the setting of Chapter 1:

- $\mathcal{D} \sim \mu^{\otimes n}$ the dataset: a finite sample of size n from the true distribution μ of support $\mathcal{X}$.
- a class of functions $\mathcal{F}$.
- $\mathcal{L}(f, z)$ a loss that depends on the predictor f and the example z. For example, in a supervised learning task with input-target pair $z = (x, y)$, the loss takes the form $\mathcal{L}(f_\theta(x), y)$ with f_θ a neural network.

The define the empirical risk as:

$$\mathcal{R}(f, \mathcal{D}) := \mathbb{E}_{z\sim\mathcal{D}}[\mathcal{L}(f, z)] = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f, z_i). \tag{8.9}$$

Note that $\hat{f}(\mathcal{D})$ is a random variable as $\mathcal{D}$ is itself a random variable. We define the population risk defined as

$$\mathcal{R}(f, \mu) := \mathbb{E}_{z\sim\mathcal{D}}[\mathcal{L}(f, z)], \tag{8.10}$$

[4]The new fashion being deep learning.

which is *not* a random variable. The generalization error is the random variable

$$\text{gen}(f, \mathcal{D}, \mu) := |\mathcal{R}(f, \mathcal{D}) - \mathcal{R}(f, \mu)|. \tag{8.11}$$

We will consider the simplest case of the Empirical Risk Minimizer (ERM):

$$\begin{aligned} f^*(\mathcal{D}) &\in \arg\inf_{f \in \mathcal{F}} \mathcal{R}(f, \mathcal{D}) \\ \text{gen}_{\text{ERM}}(\mathcal{D}, \mu) &:= \text{gen}(f^*(\mathcal{D}), \mathcal{D}, \mu). \end{aligned} \tag{8.12}$$

We will also consider the case where we have $\mathcal{A} : \mathcal{X}^n \times \mathcal{S} \to \mathcal{F}$ an algorithm that takes the dataset $\mathcal{D}$ and a random seed $s \in \S$ in input, and that returns some $\tilde{f} \in \mathcal{F}$ after a finite time T[5].

$$\begin{aligned} \tilde{f}(\mathcal{D}) &\sim \mathcal{A}(\mathcal{D}, s) \text{ with } s \sim \mathcal{U}([0,1]) \\ \text{gen}_{\mathcal{A}}(\mathcal{D}, \mu) &:= \text{gen}(\tilde{f}(\mathcal{D}), \mathcal{D}, \mu) \end{aligned} \tag{8.13}$$

Finally, we will discuss a sometimes overlooked aspect of these formulations: the domain $\mathcal{X}$ of μ and on which the elements of $\mathcal{F}$ are defined.

8.3.1 Generalization results based on function class

The typical strategy to show generalization is to show the statistical consistency of the class of functions $\mathcal{F}_{\mathcal{L}} := \{\mathcal{L} \circ f, f \in \mathcal{F}\}$, akin to the work done in chapter 4. If $\mathcal{F}_{\mathcal{L}}$ is shown to be *Glivenko-Cantelli* (A.W. van der vaart, 1996) then the empirical risk minimizer generalizes. Furthermore, if this class is shown to be *Donsker* (once again, refer to A.W. van der vaart (1996)), rates can be given on the speed at which $\text{gen}(f^*(\mathcal{D}), \mathcal{D}, \mu)$ approaches zero when $n \to +infty$. Note that in general, little can be said about the sequence $f^*(\mathcal{D})$ itself since the empirical risk minimizers are generally not unique.

All these techniques rely on the same strategy: showing that the function class $\mathcal{F}'$ is not "too big" for a well-chosen complexity measure. In the book A.W. van der vaart (1996) a large class of such complexity measures are presented. For example, the VC-dimension (see definition 13 in chapter 4) measures the largest set on which the function can fit arbitrary boolean assignments. Since it is measured as "the largest set" it is the worst case scenario for the variance (and the best-case for bias!) over all datasets $\mathcal{D} \sim \mu^{\otimes n}$. *Covering numbers* and *fat-shattering sets* are a similar construction for other classes of functions that do not take values in $\{0, 1\}$.[6]

Similarly to VC dimension, the Rademacher complexity (Koltchinskii and Panchenko, 2000) averages over random boolean assignments (using the Rademacher distribution, which is uniform over $\{-1, 1\}$), but with the advantage of also taking the expectation over $\mathcal{D} \sim \mu^{\otimes n}$. Therefore it is more flexible as it characterizes an average case, not the worst case.

PAC Bayesian framework (Shawe-Taylor and Williamson, 1997; McAllester, 1998; Dziugaite and Roy, 2017) builds upon the previous tools and adds another layer of flexibility by considering a prior over $\mathcal{F}$. This results in bounds that are very tight. According to Alquier (2021) the mutual information bounds presented in the next section are a re-discovery of previous techniques developed for PAC Bayesian learning.

[5] Contrary to most textbooks, we won't enforce any constraint on stopping time T. It can be a random variable that depends on $\mathcal{D}$ and s, and its expectation is allowed to grow faster than a polynomial in sample size n.

[6] In this regard, some theoretical results of Chapter 4 are straightforward and naive. The singularity is that I re-discovered them before realizing a theoretical framework already existed. They will serve as a testament to my growth in this area.

8.3.2 Generalization results based on algorithms

The framework of algorithmic stability introduced by Bousquet and Elisseeff (2002) focuses not on the function class $\mathcal{F}$, but rather on the property of the *algorithm* $\mathcal{A}$ when its input is perturbed: what would have been the output of $\mathcal{A}$ on a dataset $\mathcal{D}' = \mathcal{D} - \{z_i\} + \{z_i'\}$ where a single element differs? If so the output of $\mathcal{A}$ does not change too much, then $\mathcal{A}$ generalizes. This is very similar to the sensitivity introduced in Definition 19, which suggests that algorithms trained with DP guarantees must generalize - and this is indeed the case as proven in Wang et al. (2016); Jung et al. (2019). Similarly, the robustness of the algorithmic procedure implies generalization (Kawaguchi et al., 2022).

As mentioned before, in recent years, a new family of results dubbed *Mutual information Bounds* provides bounds that link the mutual information between the output hypothesis $f \sim \mathcal{A}$ with the input dataset $\mathcal{D}$. Intuitively, if this mutual information is not too high, that means that the output of $\mathcal{A}$ is somewhat decorrelated from $\mathcal{D}$, which prevents overfitting and yields generalization. This is covered in the work of Xu and Raginsky (2017) and Bu et al. (2020).

8.3.3 Generalization in countable discrete spaces for combinatorial problems

In my view, there is an issue shared by the common frameworks that theorize generalization. Most theorems assume some strong properties of the support of $\mathcal{X}$ of μ, on which the functions $\mathcal{F}$ are defined, either directly or indirectly. Typically, $\mathcal{X}$ will be assumed to be compact or bounded - which is reasonably the case for computer vision tasks, signal/speech processing, and tabular data. This hypothesis is often necessary: for example, the set of K-Lipschitz functions over $\mathcal{X}$ is not Glivenko-Cantelli on $\mathbb{R}^d$, unless we assume that $\mathcal{X}$ is also compact. Also frequently, the hypothesis that $\mathcal{X}$ is bounded will be dropped and replaced with the weaker assumption that μ is sub-gaussian. Essentially, even if the support of μ is now infinite, most of its mass will be concentrated in a small domain.

The intuition between most of the arguments is that the empirical measure $\frac{1}{n}\sum_{i=1}^{n}\delta_{z_i}$ will "fill" the space $\mathcal{X}$ fast enough, and that by interpolating on this empirical we will interpolate over the population μ. If $\mathcal{X}$ is allowed to be unbounded, and if μ is heavy-tailed, either a lot of theorems don't apply, or the right-hand side becomes vacuous. There exist some works attempting to tackle the heavy-tail regime, like Simsekli et al. (2020), but in restrictive cases like just assuming heavy-tailed gradient in SGD.

Do we care about the heavy tail regime? Yes, because the bounded/sub-gaussian regime doesn't capture properly an important class of problems: the ones operating on discrete countable spaces. This covers *all* the tasks given to a Large Language Model, which operate on *variable length sequences* defined over a discrete alphabet Σ. Better, *all* the tasks traditionally studied in complexity and calculability theory depart from decision problems[7] defined over the infinite countable space Σ^*. This covers all the problems we often care about, for example solving combinatorial tasks, finding the shortest path, finding new theorems, designing a new plane wing, finding an optimal strategy for a game, etc.

All these tasks occupy a large part of the tertiary sector of our society. Reasoning tasks fall within the framework of *Proof complexity*. For example, showing the existence of a propositional proof system that admits polynomial size proofs for all tautologies is still an open problem. This implies that reasoning (aka finding a proof) is a hard task. This is the kind of task that some

[7] Which can be seen as a binary classification task.

practitioners have been trying to teach to LLM, with the "chain-of-thoughts" reasoning introduced in Wang et al. (2022a), see also Creswell et al. (2022); Wei et al. (2022); Pei et al. (2023).

But LLM don't operate on the discrete sequence. Don't they map the tokens in a feature space $\mathbb{R}^d$? Indeed. But the tokenizer (Vaswani et al., 2017) maps a single token to a feature vector, which is different from mapping the whole sequence. To apply the classical generalization frameworks, we need to embed every word $w \in \Sigma^*$ into $\mathcal{X} \subset \mathbb{R}^d$. Two strategies can be distinguished:

1. Ensure that each encoding $\phi(w) \in \mathbb{R}^d$ is at least ϵ-apart from $\phi(w)$. Therefore, this requires an ϵ-covering of $\mathcal{X}$. The number of balls required typically grows as $\propto (\frac{1}{\epsilon})^d \text{Vol}(\mathcal{X})$ (neglecting various multiplicative factors), while the number of words in Σ^l growths as $|\Sigma^{l+1}|-1$. Therefore, solving $(\frac{1}{\epsilon})^d \text{Vol}(\mathcal{X}) \geq |\Sigma^{l+1}| - 1$ yields that $\text{Vol}(\mathcal{X}) \approx \epsilon^d |\Sigma^l|$. The volume of $\mathcal{X}$ grows quickly. For example, if $\mathcal{X}$ is chosen to be a ball, then its radius growths as $\sqrt[d]{\text{Vol}(\mathcal{X})} \approx \epsilon |\Sigma|^{l/d}$. Impossible to enforce the boundedness of $\mathcal{X}$ without restricting the hypothesis space on sequences of finite lengths l. allowing infinite growth of $\mathcal{X}$ with sub-gaussian constraint on μ would be equivalent to assigning a very small probability of very small words of Σ^*.
2. The second strategy is to choose a compact support $\mathcal{X}$. Per the previous argument, the encodings $\phi(w)$ must be arbitrarily close to each other. Therefore, the function space must contain hypotheses able to distinguish between vectors of $\mathbb{R}^d$ that are arbitrarily close to each other. This seriously threatens the smoothness properties that are required to "bound" the expressiveness of $\mathcal{F}$.

Therefore, either the support is infinite with heavy-tailed μ, making classical results inapplicable or typical bounds vacuous. Either a bounded support, but in this case the function class should be extremely expressive and distinguish between vectors that are arbitrarily close to each other, necessitating for example high Lipschitz constant, making the function space "too big" for Glivenko-Cantelli theorem to apply (variance too high). Finally, the last possibility is to enforce the huge constraint on $\mathcal{F}$ anyway, in which case the hypothesis $f \in \mathcal{F}$ might not to able to distinguish between some words. This makes the function space incapable of answering some decision problems whose input word $w \in \Sigma^*$ is too long (bias too high). Even if the error e induced by this bias is low, this small value of e can be misleading on what's happening, since this effectively translates into the incapacity to solve a countable infinite number of instances. Indeed $\sum_{w \in |\Sigma^*|} p(w) = 1$[8] regardless of the ordering, and the same can be said about $\sum_{w \in |\Sigma^*|} \mathbb{1}_{f(x)=y} p(w) = 1 - e$ the probability of the hypothesis $f \in \mathcal{F}$ to be correct. This is not a desirable property: for the shortest-path problem, for example, after showing a few solved instances to a computer science student, every professor expects him to implement Dijkstra's algorithm, which can be then run on instances of arbitrary sizes.

Remark 8.3. No free lunch for the embedding of discrete sequences?

This question of the encoding of real numbers is ubiquitous in complexity theory. The difficulty to represent and operate on real numbers is not only a limitation of Turing machines but actually, a general property that applies to more exotic computation systems. For example, the General Purpose Analog Computer (GPAC) widely discussed in Pouly (2015) corresponds to ODE of the form:

$$y'(t) = p(y(t)) \tag{8.14}$$

with $p_i : \mathbb{R}^d \to \mathbb{R}$ polynomial functions, and $y : \mathbb{R} \to \mathbb{R}^d$ the solution of the ODE for initial value $y(0) = y_0$. These systems are Turing-complete, as they can simulate a Turing machine.

[8] Sums of positive elements are absolutely convergent.

The recent work of Bournez et al. (2017) shows that problems of the class P corresponded to curves of polynomial length in the GPAC. Similarly, the recent work of Bournez et al. (2023) shows that the problems of the class PSPACE correspond to curves in the GPAC that are bounded by a polynomial. If we think to the amplitude of these curves as energy (e.g the electric current in the computation system). A careful examination of the proofs involved in these papers shows the question of the encoding $\phi(w)$ arising as a central component.

In this next section, we present an alternative framework that may circumvent existing issues.

8.3.4 Kolmogorov complexity and Levin universal search

Kolmogorov complexity (Chaitin, 1977) can be interpreted as the formal counterpart of the "Occam razor" principle that states that the best explanation or hypothesis is often the simplest. This also echoes a saying attributed to Einstein: "Everything should be made as simple as possible, but not simpler.". The Kolmogorov complexity $\mathcal{K}(s_n)$ of the sequence $s_n \in \Sigma^n$ of length n is the length of the *shortest* program P that output s_n. We recall below some interesting properties.

Exemple 8.3. Kolmogorov complexity of a few sequences.

The sequence $[0, 2, 4, 6, 8, 10, 12]$, is produced by the simple for loop that output $2i$ for $i \in [0, 6]$. Similarly, $[1, 2, 4, 8, 16, 32, 64, ...]$ is a for loop that outputs 2^i. Even the sequence $[3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, ...]$ is structured: the elements can be produced by a program computing the digits of π. These infinite sequences admit a finite description as a program.

The notion of "shortest" program implies a choice of a particular machine (e.g Turing machine, Python, C++?). But since every Turing complete system can simulate any other, all these quantities are related up to a constant. For example, if $\mathcal{K}(s_n)$ is the Kolmogorov complexity of s_n in C++, if $\mathcal{K}'(s_n)$ is the Kolmogorov complexity of s_n in Python, and if $C(\mathcal{K}, \mathcal{K}')$ is the length of a Python interpreter described in C++, then the inequality $\mathcal{K}(s_n) \leq \mathcal{K}'(s_n) + C(\mathcal{K}, \mathcal{K}')$ holds: it is enough to simulate Python in C++ and use the shortest program available for Python. The reverse inequality $\mathcal{K}'(s_n) \leq \mathcal{K}(s_n) + C(\mathcal{K}', \mathcal{K})$ holds (remember that $C(\cdot, \cdot)$ has no reason to be symmetric). A sequence s_n will be qualified as random if $\mathcal{K}(s_n)$ growths as $\mathcal{O}(n)$. As $C(\cdot, \cdot)$ does not depend on n, when n grows to $+\infty$, its value becomes negligible. Therefore, if a sequence is random for a given computation model $\mathcal{K}(\cdot)$, it is for every model. Randomness is an intrinsic property of the sequence, a fundamental lack of structure that prevents it from being accurately captured by programs. Conversely, this suggests that "structure" is a universal notion, which motivated a "algorithmic theory of everything" based on this unifying principle (Schmidhuber, 2000).

A related terminology, while being a bit less precise, is the *minimum description length principle* (Grünwald, 2007). While most generalization theory takes the statistical viewpoint of "interpolation", by filling the space $\mathcal{X}$ with the empirical measure, the Kolmogorov complexity opens a path towards "extrapolation" by considering sequences of arbitrary length, *and by assuming that the sequences are highly structured*. This last hypothesis is not easily translatable in hypothesis on the measure μ, which may explain why the current framework struggles to capture this property. Maybe we are asking too much from our machine learning algorithms, and maybe we don't about generalizing for arbitrary measures μ: maybe all we care about is generalizing *arbitrarily well* for some specific measures μ.

These ideas have been repeatedly studied in deep learning, starting with Pearlmutter and Rosenfeld (1990) and Schmidhuber (1997), and got a surge of interest in the recent years, for example

with the works of Blier and Ollivier (2018) and Lee et al. (2022).

The Kolmogorov complexity has an issue: it is not computable (in the sense of Turing). Therefore, while appealing, this tool has no practical interest. However, a small modification yields a related quantity, *which is* computable. Inspired by the Levin universal search algorithm (Levin, 1973, 1984), this quantity *regularized* the length of the program by the associated runtime (Li et al., 2008):

$$\mathcal{K}_t(s_n) := \min_{P:s_n \sim P} l(P) + \log t(P), \tag{8.15}$$

where P is a program that output s_n, $t(P)$ is the time needed to run P, and $l(P)$ the length of P. By regularizing with computation time, it is possible to run multiple programs in parallel and stop whichever output s_n first. The runtime is a reasonable penalty, as a short program that takes forever to finish is of little practical use.

This idea of computational constraint also appeared in another context, to define a generalization of the Shannon entropy: the ν-information (Xu et al., 2019). It has been applied to characterize the complexity of a dataset, in a scheme that reminds of Kolmogorov complexity (Ethayarajh et al., 2022). Unlike Shannon entropy, which only depends on the joint distribution of variables, ν-information assumes that the decoder has limited decoding capabilities, and may be unable to distinguish between some messages. The opportunity is too good not to cite Levin (2003): "From time immemorial, humanity has got frequent, often cruel, reminders that many things are easier to do than to reverse. When the foundations of mathematics started to be seriously analyzed, this experience immediately found a formal expression". These *one-way functions*, often used in modern cryptography, are invertible functions for which computing the forward map $x \mapsto h(x) = y$ is "easy" (runtime polynomially bounded) but reversing them as $y \mapsto h^{-1}(y) = x$ is "hard" (no better algorithm that enumerating every x and performing the forward). For these functions the ν-mutual information is written as $I_\nu(X \to Y)$. In this case, when ν represent the computation capabilities of the decoder, and if ν is reasonably bounded, one should expect that $I_\nu(X \to h(X)) \gg I_\nu(h(X) \to X)$ since encoding with h is easier than decoding. In my opinion, these concepts offer a fruitful point of view to theorize generalization in these discrete spaces.

8.4 So, when is AGI coming, then?

In this section, we attempt to define the contours of Artificial General Intelligence (AGI), as often discussed in the media. We will review different possible definitions. We will conclude on the practical feasibility of these definitions.

8.4.1 What is AGI?

At the time of writing (2023), there is a general consensus in the population and scientific community that generative models, and especially Large Language Models, are what we have closest to an AGI. As chatbots, they take into account the context, they handle uncertainties and ambiguities in human language. They can also reason on some hypotheses (even if they are sometimes wrong). They succeed as many evaluations created for humans and are considered challenging (e.g the bar exam as shown in Katz et al. (2023)). They shine in translation tasks, sometimes being able to translate idioms or explain jokes. They are also good at summarizing content or analyzing emotions and tone (positive, negative). Yet, they are not considered as AGI yet. They are still *hallucinating* (Ji et al., 2023), and are not capable (yet) of self-improvement on new tasks.

In some literature, AGI is understood as some sort of entity with almost infinite computation power, which could be fed any problem, and provide in few seconds meaningful answers and solutions.

Such a system would be similar to Laplace's demon, as described in "We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past could be present before its eyes." (Laplace, 1835).

In other more pragmatic literature, AGI is understood as a system with capabilities similar to humans, or superior, but of a similar nature anyway. The system would adapt itself to new situations in record times (with low *sample complexity*), be able to perform complex reasoning, distinguish truth from hallucinations, and take initiative.

We extrapolate below the requirements of these two systems.

8.4.2 The unreasonable energy requirements of "god-level" AGI

This form of AGI is unlikely to exist ever since it would violate several laws of the complexity theory and thermodynamics.

Complexity theory. states that some tasks cannot be solved exactly, or even approximately, without a lower bound on the computation time and the memory requirements. For example, if $P \neq NP$ it is likely that solving exactly some instances of NP-hard problems would require an exponential amount of computations. And a lot of problems are actually harder than NP-complete: finding the optimal strategy in a deterministic two-player game is already PSPACE-complete. Deciding if the computation of a system halts after k steps is EXPTIME-complete. Deciding if two regular expressions are equivalent is EXPSPACE-complete. All these rather artificial problems are abstractions of more frequent problems that can be encountered by an intelligent system. Even in the very optimistic case where $P = NP = PSPACE$, it is already known that $P \subsetneq EXPTIME$ and $PSPACE \subsetneq EXPSPACE$: the complexity classes do not collapse (Arora and Barak, 2009). All these considerations would be a theorist problem if it wasn't for the *Landauer's principle*.

Landauer's principle. Landauer's principle (Landauer, 1961) states that the minimum energy needed to erase one bit of information is proportional to the temperature at which the system is operating. This reads as

$$E \geq k_B T \ln(2), \tag{8.16}$$

with E the energy loss per bit erased, k_B the Boltzmann constant, and T the temperature. This inequality arises a solution to *Maxwell's demon paradox* (Thomson, 1875): an entity that could be able to predict the position and the velocity of every particle in two rooms connected, could selectively open and close the doors between the rooms to concentrate cold particles on one side, and warm particles to the other. In appearance, this would violate the second principle of thermodynamics, by transforming a system with two rooms at equal temperatures, into one with two different temperatures, from which *work* could be extracted. The paradox is solved because such a demon would need to process information, and information processing is generally a non-reversible operation, and non-reversible transformation must dissipate energy.

Exemple 8.4. Practical example.

We assume we have at our disposal an infinitely efficient computer, operating at the temperature of cosmological background (3 Kelvins), which costs about 10^{-23} Joules per bit erased. We assume that we are trying to solve some hard instance of EXPTIME complete problem that

requires rewriting 10^n bits for an instance of size n. We assume that the total energy produced on earth (including electricity, oil engines, etc.) in 2022 is fed to this efficient computer, which is about 3,112 million tons of equivalent petrol, which is about 10^{20} Joules in total. This computer can erase about 10^{43} bits before running out of energy. This effectively limits the maximum size of an instance to $n = \log(10^{43}) = \ldots 43$. Of course, this neglects the additional energy required to evacuate the heat and maintain the system at 3 Kelvins.

The Landauer's limit can be avoided with *reversible computations*, which arises in quantum circuits for example (Aaronson, 2013). Unfortunately, this implies that the size of the system must grow with its memory since no information can be lost. The energy of a system is directly proportional to its energy, and we fall back to the previous issue: either the system dissipates enormous amounts of energy to re-use memory slots (by erasing and rewriting), either the system greedily consumes mass and energy to store information and ensures reversibility of the computation.

Plancks limits. In some theory the universe admits a minimal measurable length: the Planck length. Similarly, it admits a maximum energy density. As energy and information are linked, the universe admits a maximum information density. The Bremermann's limit (Bremermann et al., 1962) is the maximum rate of computation achievable per a physical system, and its value is around 10^{50} bits per second per kilogram.

Overall, these different results suggest that implementing a "god level" AGI that can reason fast, discover theorems, and solve complex combinatorics problems is an ill-posed problem, that will never be physically possible, and we should lower our expectations[9].

8.4.3 The more reasonable AGI: human look-alike

In this setting, we lower our expectations to an AGI whose learning capabilities are on par with humans, with similar creativity, able to recognize new situations, and learn "efficiently" (with low sample complexity) new tasks, on rich multimodal inputs. What is lacking for this AGI to exist? Probably:

1. a memory and a variable-length computation mechanism.
2. "embodiment": the rich sensorial and multimodal input of a body that interacts with the world.

Here, we focus the discussion on memory and variable-length computation mechanisms.

Memory in LLM

Today, in LLM two memories co-exist.

1. The first memory is embedded in the weights θ of the network. It compresses most of the knowledge contained in the training corpus[10]. The issue with this memory is that it is static: it cannot be updated easily when the agent interacts with the world. Updating this knowledge would typically require fine-tuning and tools from continual learning field (Parisi et al., 2019), since neural networks are known to suffer from *catastrophic forgetting* (Kirkpatrick et al., 2017): by learning new tasks if one if not careful, previous tasks may be forgotten, *even when the*

[9] Complexity theory has always been a horribly depressing field, and neural networks cannot change that.

[10] To some extent, intelligence *is* compression. If you have a minute or two to kill, I happily suggest that you take a look at `https://www.youtube.com/watch?v=ubNF9QNEQLA`. The phenomenon of this video is often presented as related to *awareness*. I propose an alternative hypothesis: the scene understanding is compressed by the brain. Visually, the two scenes are identical if we use textual description as a compression: a detective, policeman, three suspects, a dead, a carpet, old fashion mansion decoration. The differences are not spotted because the visual input is compressed into the same description.

support $\mathcal{X}$ of these tasks is disjoint. Therefore, the weights θ of the network are typically kept frozen during interactions with the user. Overall, this is akin to the long-term memory of humans and similar to the role of a hard drive in computers.

2. The second memory is provided by *the context*: the history of previous interactions by the user. This is what allows *chain of thoughts* reasoning in LLM. The size of this memory is limited by the number of tokens that the model can support. Overall, this is akin to the "short-term memory" of humans and similar to the role of RAM in computers. Note that even in Vision Transformer, the transformers benefit from additional tokens to carry-out computations, as suggested in the recent work of Darcet et al. (2023).

It seems that a third form of memory is required for the system to be complete. In the current state, there is no way to integrate the results of short-term memory into long memory. However, it has been known since the *Von Neumann* architecture (Von Neumann, 1945) that allowing the *program* (here the neural network architecture) and the *data* (here, the context tokens) to co-exist in the place allows the treat instructions as data. This opens the path to self-improvement of the system, which can operate on its own source code, and is the basis for compilers or computer viruses.

Furthermore, complexity theory suggests, that for hard tasks the amount of memory should be allowed to grow arbitrarily high, which requires unbounded contexts, or the capacity to query an external memory for reads and writes. This additional property makes the LLM truly Turing-complete, as argued in Schuurmans (2023).

Variable-length computation mechanisms

Complexity theory also shows that complex tasks require a computation budget that must scale with the difficulty of the task, where "difficulty" encompasses both the nature of the problem and the size of the instance. Therefore, future learning will need to implement a "while loop" mechanism. The sampler in LLM is an example of a solution: at each step, the LLM assigns a probability to the new token, among which a special STOP token[11] stops the production of the answer.

Self-modifying capabilities based on previous computations, and arbitrarily long computation times, are central tools for self-improvement, which is the basis of intelligence.

8.5 Conclusion to the conclusion

Lipschitz networks are a fascinating lens through which to analyze and understand the dynamics of learning in neural networks. They show promises for robust and safe learning. Here, robustness is better understood in a broad sense that covers robustness against adversarial attacks, with privacy guarantees, and with generalization guarantees. Some of the works drafted suggest that they offer tools for calibration, or reliability for neural signed distance functions rendering. Their construction and their training rely entirely on a careful examination of the computation graph and back-propagation algorithm, which emphasizes the importance of the deep learning paradigm. Finally, it appears that Lipschitz constraints alone are probably insufficient to reach AGI and scale neural network to complex combinatorial tasks that requires long chains of reasoning, but may be worth taking into consideration to design stable training algorithms.

[11] This is a "break" statement in programming languages.

Bibliography

S. Aaronson. *Quantum computing since Democritus*. Cambridge University Press, 2013.

M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

P. Ablin and G. Peyré. Fast and accurate optimization on the orthogonal manifold without retraction. In *International Conference on Artificial Intelligence and Statistics*, pages 5636–5657. PMLR, 2022.

P.-A. Absil and J. Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135–158, 2012.

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning Representations and Generative Models for 3D Point Clouds, June 2018.

E. M. Achour, F. Malgouyres, and F. Mamalet. Existence, stability and scalability of orthogonal convolutional neural networks. *The Journal of Machine Learning Research*, 23(1):15743–15798, 2022.

Airbus. Decomon. `https://github.com/airbus/decomon`, 2023.

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

F. Alimisis, A. Orvieto, G. Becigneul, and A. Lucchi. Momentum improves optimization on riemannian manifolds. In *International conference on artificial intelligence and statistics*, pages 1351–1359. PMLR, 2021.

P. Alquier. User-friendly introduction to pac-bayes bounds. *arXiv preprint arXiv:2110.11216*, 2021.

J. Altschuler, J. Niles-Weed, and P. Rigollet. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Advances in neural information processing systems*, pages 1964–1974, 2017.

M. S. Alvim, M. E. Andrés, K. Chatzikokolakis, P. Degano, and C. Palamidessi. Differential privacy: on the trade-off between utility and information leakage. In *Formal Aspects of Security and Trust: 8th International Workshop, FAST 2011, Leuven, Belgium, September 12-14, 2011. Revised Selected Papers 8*, pages 39–54. Springer, 2012.

B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

B. Amos, L. Xu, and J. Z. Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

B. Amos et al. Tutorial on amortized optimization. *Foundations and Trends® in Machine Learning*, 16(5):592–732, 2023.

G. Andrew, O. Thakkar, B. McMahan, and S. Ramaswamy. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems*, 34:17455–17466, 2021.

C. Anil, J. Lucas, and R. Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pages 291–301. PMLR, 2019.

A. Araujo, B. Negrevergne, Y. Chevaleyre, and J. Atif. Training compact deep learning models for video classification using circulant matrices. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

A. Araujo, B. Negrevergne, Y. Chevaleyre, and J. Atif. Understanding and training deep diagonal circulant neural networks. *arXiv preprint arXiv:1901.10255*, 2019.

A. Araujo, B. Negrevergne, Y. Chevaleyre, and J. Atif. On lipschitz regularization of convolutional layers using toeplitz matrix theory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6661–6669, 2021.

A. Araujo, A. J. Havens, B. Delattre, A. Allauzen, and B. Hu. A unified algebraic perspective on lipschitz neural networks. In *The Eleventh International Conference on Learning Representations*, 2022.

M. Arjovsky, A. Shah, and Y. Bengio. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128. PMLR, 2016.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

L. Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

M. Atzmon and Y. Lipman. SAL: Sign Agnostic Learning of Shapes from Raw Data, Mar. 2020.

J. W. A.W. van der vaart. *Weak Convergence and Empirical Processes*. Springer-Verlag New York, 1996.

M. Ayara, J. Timmis, R. Lemos, L. De Castro, and R. Duncan. Negative selection: How to generate detectors. *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, Jan. 2002.

M. Azizmalayeri, A. S. Moakar, A. Zarei, R. Zohrabi, M. T. Manzuri, and M. H. Rohban. Your out-of-distribution detection method is not robust! In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

F. Bachoc, A. Suvorikova, D. Ginsbourger, J.-M. Loubes, and V. Spokoiny. Gaussian processes with multidimensional distribution inputs via optimal transport and Hilbertian embedding. *Electronic journal of statistics*, 14(2):2742–2772, 2020.

F. Bachoc, L. Béthune, A. Gonzalez-Sanz, and J.-M. Loubes. Gaussian processes on distributions based on regularized optimal transport. In *International Conference on Artificial Intelligence and Statistics*, pages 4986–5010. PMLR, 2023a.

F. Bachoc, L. Béthune, A. González-Sanz, and J.-M. Loubes. Improved learning theory for kernel distribution regression with two-stage sampling. *arXiv preprint arXiv:2308.14335*, 2023b.

S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang. Recent advances in adversarial training for adversarial robustness. In Z.-H. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4312–4321. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/591. Survey Track.

G. A. Baker Jr and J. L. Gammel. The padé approximant. *Journal of Mathematical Analysis and Applications*, 2(1):21–30, 1961.

C. Bálint, G. Valasek, and L. Gergó. Operations on signed distance function estimates. *preprint*, 2023.

B. Balle, G. Barthe, and M. Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. *Advances in Neural Information Processing Systems*, 31, 2018.

S. Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta mathematicae*, 3(1):133–181, 1922.

G. Banjac, P. Goulart, B. Stellato, and S. Boyd. Infeasibility detection in the alternating direction method of multipliers for convex optimization. *Journal of Optimization Theory and Applications*, 183(2):490–519, 2019. doi: 10.1007/s10957-019-01575-y.

N. Bansal, X. Chen, and Z. Wang. Can we gain more from orthogonality regularizations in training deep networks? *Advances in Neural Information Processing Systems*, 31, 2018.

P. Bartlett. For valid generalization the size of the weights is more important than the size of the network. *Advances in neural information processing systems*, 9, 1996.

P. L. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6241–6250, 2017.

P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20:63–1, 2019.

J. Beirlant, S. Buitendag, E. Del Barrio, M. Hallin, and F. Kamper. Center-outward quantiles and the measurement of multivariate risk. *Insurance: Mathematics and Economics*, 95:79–100, 2020.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32): 15849–15854, 2019.

B. M. Bell and J. V. Burke. Algorithmic differentiation of implicit functions and optimal values. In *Advances in Automatic Differentiation*, pages 67–77. Springer, 2008.

H. Ben-Hamu, H. Maron, I. Kezurer, G. Avineri, and Y. Lipman. Multi-chart Generative Surface Modeling. *ACM Transactions on Graphics*, 37(6):1–15, Dec. 2018. ISSN 0730-0301, 1557-7368. doi: 10.1145/3272127.3275052.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

B. Bercu, J. Bigot, and G. Thurin. Monge-kantorovich superquantiles and expected shortfalls with applications to multivariate risk measurements. *arXiv e-prints*, pages arXiv–2307, 2023.

L. Bergman and Y. Hoshen. Classification-based anomaly detection for general data. In *International Conference on Learning Representations*, 2019.

Q. Bertrand and M. Massias. Anderson acceleration of coordinate descent. In *International Conference on Artificial Intelligence and Statistics*, pages 1288–1296. PMLR, 2021.

L. Béthune, T. Boissin, M. Serrurier, F. Mamalet, C. Friedrich, and A. G. Sanz. Pay attention to your loss : understanding misconceptions about lipschitz neural networks. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

L. Béthune, P. Novello, G. Coiffier, T. Boissin, M. Serrurier, Q. Vincenot, and A. Troya-Galvis. Robust one-class classification with signed distance function using 1-Lipschitz neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2245–2271. PMLR, 23–29 Jul 2023.

D. A. Bini, N. J. Higham, and B. Meini. Algorithms for the matrix p th root. *Numerical Algorithms*, 39(4):349–378, 2005.

C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

J. Bitterwolf, A. Meinke, and M. Hein. Certifiably adversarially robust detection of out-of-distribution data. In *Advances in Neural Information Processing Systems*, volume 33, pages 16085–16095. Curran Associates, Inc., 2020.

Å. Björck and C. Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Journal on Numerical Analysis*, 8(2):358–364, 1971.

E. Black, S. Yeom, and M. Fredrikson. Fliptest: fairness testing via optimal transport. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 111–121, 2020.

L. Blier and Y. Ollivier. The description length of deep learning models. *Advances in Neural Information Processing Systems*, 31, 2018.

M. Blondel, A. Martins, and V. Niculae. Learning classifiers with fenchel-young losses: Generalized entropies, margins, and algorithms. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 606–615. PMLR, 2019.

M. Blondel, A. F. Martins, and V. Niculae. Learning with fenchel-young losses. *The Journal of Machine Learning Research*, 21(1):1314–1382, 2020.

M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert. Efficient and modular implicit differentiation. *Advances in neural information processing systems*, 35:5230–5242, 2022.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

V. I. Bogachev and M. A. S. Ruas. *Measure theory*, volume 1. Springer, 2007.

D. Boltcheva and B. Lévy. Surface reconstruction by computing restricted voronoi cells in parallel. *Computer-Aided Design*, 90:123–134, 2017.

J. Bolte, E. Pauwels, and S. Vaiter. Automatic differentiation of nonsmooth iterative algorithms. *Advances in Neural Information Processing Systems*, 35:26404–26417, 2022.

L. Bonicelli, M. Boschini, A. Porrello, C. Spampinato, and S. Calderara. On the effectiveness of lipschitz-driven rehearsal in continual learning. *Advances in Neural Information Processing Systems*, 35:31886–31901, 2022.

S. Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

S. Boucheron, G. Lugosi, and P. Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

O. Bournez, D. S. Graça, and A. Pouly. Polynomial time corresponds to solutions of polynomial ordinary differential equations of polynomial length. *Journal of the ACM (JACM)*, 64(6):1–76, 2017.

O. Bournez, R. Gozzi, D. S. Graça, and A. Pouly. A continuous characterization of pspace using polynomial ordinary differential equations. *Journal of Complexity*, 77:101755, 2023.

O. Bousquet and A. Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

V. Boutin, A. Zerroug, M. Jung, and T. Serre. Iterative vae as a predictive brain model for out-of-distribution generalization. In *NeurIPS 2020 Workshop*, 2020.

S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

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

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

H. J. Bremermann et al. Optimization through evolution and recombination. *Self-organizing systems*, 93:106, 1962.

M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, may 2000. ISSN 0163-5808. doi: 10.1145/335191.335388.

A. Brilhault, S. Neuenschwander, and R. A. Rios. A new robust multivariate mode estimator for eye-tracking calibration. *Behavior Research Methods*, 55(2):516–553, 2023.

R. A. Brualdi. *Combinatorial matrix classes*, volume 13. Cambridge University Press, 2006.

Y. Bu, S. Zou, and V. V. Veeravalli. Tightening mutual information-based bounds on generalization error. *IEEE Journal on Selected Areas in Information Theory*, 1(1):121–130, 2020.

Z. Bu, J. Mao, and S. Xu. Scalable and efficient training of large convolutional neural networks with differential privacy. *Advances in Neural Information Processing Systems*, 35:38305–38318, 2022a.

Z. Bu, Y.-X. Wang, S. Zha, and G. Karypis. Automatic clipping: Differentially private deep learning made easier and stronger. *arXiv preprint arXiv:2206.07136*, 2022b.

Z. Bu, Y.-X. Wang, S. Zha, and G. Karypis. Differentially private optimization on large model at small cost. In *International Conference on Machine Learning*, pages 3192–3218. PMLR, 2023.

S. Bubeck and M. Sellke. A universal law of robustness via isoperimetry. *Advances in Neural Information Processing Systems*, 34:28811–28822, 2021.

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

C. Bunne, A. Krause, and M. Cuturi. Supervised training of conditional monge maps. *Advances in Neural Information Processing Systems*, 35:6859–6872, 2022.

J. Cape, M. Tang, and C. E. Priebe. The two-to-infinity norm and singular subspace geometry with applications to high-dimensional statistics. *The Annals of Statistics*, 47(5):2405–2439, 2019.

J. Céa. Conception optimale ou identification de formes, calcul rapide de la dérivée directionnelle de la fonction coût. *M2AN-Modélisation mathématique et analyse numérique*, 20(3):371–402, 1986.

G. J. Chaitin. Algorithmic information theory. *IBM journal of research and development*, 21(4): 350–359, 1977.

A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1):25–45, 2021.

C.-Y. Chao. A remark on symmetric circulant matrices. *Linear Algebra and its Applications*, 103: 133–148, 1988.

A. Charguéraud, J.-C. Filliâtre, C. Lourenço, and M. Pereira. Gospel—providing ocaml with a formal specification language. In *Formal Methods–The Next 30 Years: Third World Congress, FM 2019, Porto, Portugal, October 7–11, 2019, Proceedings 3*, pages 484–501. Springer, 2019.

S. Chaudhari, S. Pranav, and J. M. Moura. Learning gradients of convex functions with monotone gradient networks. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

X. Chen, S. Z. Wu, and M. Hong. Understanding gradient clipping in private sgd: A geometric perspective. *Advances in Neural Information Processing Systems*, 33:13773–13782, 2020.

P. Cheridito, A. Jentzen, and F. Rossmannek. Efficient approximation of high-dimensional functions with neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7): 3079–3093, 2021.

A. Chlipala. *Certified programming with dependent types: a pragmatic introduction to the Coq proof assistant*. MIT Press, 2022.

K. Choromanski, D. Cheikhi, J. Davis, V. Likhosherstov, A. Nazaret, A. Bahamou, X. Song, M. Akarte, J. Parker-Holder, J. Bergquist, et al. Stochastic flows and geometric optimization on the orthogonal group. In *International Conference on Machine Learning*, pages 1918–1928. PMLR, 2020.

G. Chou, I. Chugunov, and F. Heide. GenSDF: Two-Stage Learning of Generalizable Signed Distance Functions, Oct. 2022.

A. Christmann and I. Steinwart. Universal kernels on non-standard input spaces. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

A. Cichocki and A.-H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 92(3):708–721, 2009.

M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier. Parseval networks: Improving robustness to adversarial examples. In *International Conference on Machine Learning*, pages 854–863. PMLR, 2017.

E. A. Coddington, N. Levinson, and T. Teichmann. Theory of ordinary differential equations, 1956.

J. Cohen, E. Rosenfeld, and Z. Kolter. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 2019.

G. Coiffier, P. Renard, and S. Lefebvre. 3d geological image synthesis from 2d examples using generative adversarial networks. *Frontiers in Water*, 2:560598, 2020.

A. Creswell, M. Shanahan, and I. Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2022.

C. Cui, Z. Yan, G. Liu, and L. Lu. Generalizing and improving jacobian and hessian regularization. *arXiv preprint arXiv:2212.00311*, 2022.

M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

M. Cuturi, L. Meng-Papaxanthos, Y. Tian, C. Bunne, G. Davis, and O. Teboul. Optimal transport tools (ott): A JAX toolbox for all things Wasserstein. *arXiv preprint arXiv:2201.12324*, 2022.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

O. Dalmau-Cedeno and H. Oviedo. A projection method for optimization problems on the stiefel manifold. In *Mexican conference on pattern recognition*, pages 84–93. Springer, 2017.

T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

R. Das, S. Kale, Z. Xu, T. Zhang, and S. Sanghavi. Beyond uniform lipschitz condition in differentially private optimization. In *International Conference on Machine Learning*, pages 7066–7101. PMLR, 2023.

G. Dasoulas, K. Scaman, and A. Virmaux. Lipschitz normalization for self-attention layers with application to graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, pages 2456–2466. PMLR, 2021.

T. Davies, D. Nowrouzezahrai, and A. Jacobson. On the Effectiveness of Weight-Encoded Neural Implicit 3D Shapes, Jan. 2021.

Dawson-Haggerty et al. trimesh, 2019. URL `https://trimsh.org/`.

S. De, L. Berrada, J. Hayes, S. L. Smith, and B. Balle. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022.

L. De Lara, A. González-Sanz, N. Asher, and J.-M. Loubes. Transport-based counterfactual models. *arXiv preprint arXiv:2108.13025*, 2021.

E. del Barrio, J. A. Cuesta-Albertos, M. Hallin, and C. Matrán. Center-outward distribution functions, quantiles, ranks, and signs. *arXiv preprint arXiv:1806.01238*, 2018.

E. del Barrio, A. G. Sanz, and M. Hallin. Nonparametric multiple-output center-outward quantile regression. *arXiv preprint arXiv:2204.11756*, 2022.

B. Delattre, A. Araujo, Q. Barthélemy, and A. Allauzen. The lipschitz-variance-margin tradeoff for enhanced randomized smoothing. *arXiv preprint arXiv:2309.16883*, 2023a.

B. Delattre, Q. Barthélemy, A. Araujo, and A. Allauzen. Efficient bound of Lipschitz constant for convolutional layers by Gram iteration. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7513–7532. PMLR, 23–29 Jul 2023b.

H. Delseny, C. Gabreau, A. Gauffriau, B. Beaudouin, L. Ponsolle, L. Alecu, H. Bonnin, B. Beltran, D. Duchel, J.-B. Ginestet, et al. White paper machine learning in certified systems. *arXiv preprint arXiv:2103.10529*, 2021.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

R. DeVore, B. Hanin, and G. Petrova. Neural network approximation. *Acta Numerica*, 30:327–444, 2021.

B. Dherin, M. Munn, M. Rosca, and D. Barrett. Why neural networks find simple solutions: The many regularizers of geometric complexity. *Advances in Neural Information Processing Systems*, 35:2333–2349, 2022.

Y. Ding, Y. Zheng, Z. Han, and X. Yang. Using optimal transport theory to optimize a deep convolutional neural network microscopic cell counting method. *Medical & Biological Engineering & Computing*, pages 1–12, 2023.

Y. Dong, H. Su, J. Zhu, and F. Bao. Towards interpretable deep neural networks by leveraging adversarial examples. *In AAAI-19 Workshop on Network Interpretability for Deep Learning*, 2017.

Y. Du and I. Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

Y. Du, S. Li, and I. Mordatch. Compositional visual generation with energy based models. *Advances in Neural Information Processing Systems*, 33:6637–6647, 2020.

Y. Du, C. Durkan, R. Strudel, J. B. Tenenbaum, S. Dieleman, R. Fergus, J. Sohl-Dickstein, A. Doucet, and W. S. Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International Conference on Machine Learning*, pages 8489–8510. PMLR, 2023.

P. Dvurechensky, A. Gasnikov, and A. Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by Sinkhorn's algorithm. In *International conference on machine learning*, pages 1367–1376. PMLR, 2018.

C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer, 2006.

C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

S. d'Ascoli, M. Refinetti, G. Biroli, and F. Krzakala. Double trouble in double descent: Bias and variance (s) in the lazy regime. In *International Conference on Machine Learning*, pages 2280–2290. PMLR, 2020.

Y. Ebihara, X. Dai, V. Magron, D. Peaucelle, and S. Tarbouriech. Local lipschitz constant computation of relu-fnns: Upper bound computation with exactness verification. *arXiv preprint arXiv:2310.11104*, 2023.

A. Ebrahimpour-Boroojeny, M. Telgarsky, and H. Sundaram. Spectrum extraction and clipping for implicitly linear layers. In *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*, 2023.

M. Eisenberger, A. Toker, L. Leal-Taixé, F. Bernard, and D. Cremers. A unified framework for implicit Sinkhorn differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 509–518, 2022.

L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, 2021.

A. El Hanchi, D. Stephens, and C. Maddison. Stochastic reweighted gradient descent. In *International Conference on Machine Learning*, pages 8359–8374. PMLR, 2022.

N. B. Erichson, O. Azencot, A. Queiruga, L. Hodgkinson, and M. W. Mahoney. Lipschitz recurrent neural networks. In *International Conference on Learning Representations*, 2021.

K. Ethayarajh, Y. Choi, and S. Swayamdipta. Understanding dataset difficulty with v-usable information. In *International Conference on Machine Learning*, pages 5988–6008. PMLR, 2022.

J. A. Evans and T. J. Hughes. Isogeometric divergence-conforming b-splines for the darcy–stokes–brinkman equations. *Mathematical Models and Methods in Applied Sciences*, 23(04):671–741, 2013.

F. Faghri, C. Vasconcelos, D. J. Fleet, F. Pedregosa, and N. L. Roux. Bridging the gap between adversarial robustness and optimization bias. *ICLR 2021 Workshop on Security and Safety in Machine Learning Systems*, 2021.

J. Fan, S. Liu, S. Ma, H.-M. Zhou, and Y. Chen. Neural monge map estimation and its applications. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. Featured Certification.

M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

T. Fel, V. Boutin, M. Moayeri, R. Cadène, L. Bethune, M. Chalvidal, T. Serre, et al. A holistic approach to unifying automatic concept extraction and concept importance estimation. *Advances in Neural Information Processing Systems*, 2023a.

T. Fel, A. Picard, L. Bethune, T. Boissin, D. Vigouroux, J. Colin, R. Cadène, and T. Serre. Craft: Concept recursive activation factorization for explainability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2711–2721, 2023b.

C. Févotte and J. Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural computation*, 23(9):2421–2456, 2011.

A. Figalli. On the continuity of center-outward distribution and quantile functions. *Nonlinear Analysis*, 177:413–421, 2018.

C. Finlay, J. Calder, B. Abbasi, and A. Oberman. Lipschitz regularized deep neural networks generalize and are adversarially robust. *arXiv preprint arXiv:1808.09540*, 2018.

R. Flamary and N. Courty. POT Python optimal transport library, 2017. URL `https://github.com/rflamary/POT`.

S. Flaxman, D. J. Sutherland, Y.-X. Wang, and Y. W. Teh. Understanding the 2016 us presidential election using ecological inference and distribution regression with census microdata. *arXiv preprint arXiv:1611.03787*, 2016.

S. R. Flaxman, Y.-X. Wang, and A. J. Smola. Who supported obama in 2012? ecological inference through distribution regression. In *International Conference on Knowledge Discovery and Data Mining*, 2015.

M. Fontana, G. Zeni, and S. Vantini. Conformal prediction: a unified review of theory and new challenges. *arXiv preprint arXiv:2005.07972*, 2020.

S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proceedings of 1994 IEEE computer society symposium on research in security and privacy*, pages 202–212. Ieee, 1994.

S. Fort, J. Ren, and B. Lakshminarayanan. Exploring the limits of out-of-distribution detection. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

J. H. Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4): 367–378, 2002.

S. Fu, N. Tamir, S. Sundaram, L. Chai, R. Zhang, T. Dekel, and P. Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data, 2023.

S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin. Jfb: Jacobian-free backpropagation for implicit networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6648–6656, 2022.

W. Gangbo and R. J. McCann. The geometry of optimal transportation. *Acta Mathematica*, 177: 113–161, 1996.

Y. Gao, M. K. Ng, and M. Zhou. Approximating probability distributions by using wasserstein generative adversarial networks. *SIAM Journal on Mathematics of Data Science*, 5(4):949–976, 2023.

A. Genevay, G. Peyré, and M. Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617. PMLR, 2018.

F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. In *Ninth International Conference on Artificial Neural Networks ICANN 99*. IET, 1999.

S. Ghazanfari, A. Araujo, P. Krishnamurthy, F. Khorrami, and S. Garg. Lipsim: A provably robust perceptual similarity metric. *arXiv preprint arXiv:2310.18274*, 2023.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

F. Gogianu, T. Berariu, M. Rosca, C. Clopath, L. Busoniu, and R. Pascanu. Spectral normalization for deep reinforcement learning: an optimisation perspective. In *Proceedings of the International Conference on Machine Learning (ICML)*. JMLR. org, 2021.

I. Golan and R. El-Yaniv. Deep anomaly detection using geometric transformations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM computing surveys (CSUR)*, 23(1):5–48, 1991.

A. Golinski, M. Lezcano-Casado, and T. Rainforth. Improving normalizing flows via better orthogonal parameterizations. In *First workshop on Invertible Neural Networks and Normalizing Flows (ICML)*, 2019.

F. Gonzalez, D. Dasgupta, and R. Kozma. Combining negative selection and classification techniques for anomaly detection. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 1, pages 705–710. IEEE, 2002.

A. González-Sanz, L. De Lara, L. Béthune, and J.-M. Loubes. Gan estimation of lipschitz optimal transport maps. *arXiv preprint arXiv:2202.07965*, 2022.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

L.-A. Gottlieb, A. Kontorovich, and R. Krauthgamer. Efficient classification for metric data. *IEEE Transactions on Information Theory*, 60(9):5750–5759, 2014.

H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110:393–416, 2021.

S. Goyal, A. Raghunathan, M. Jain, H. V. Simhadri, and P. Jain. DROCC: Deep Robust One-Class Classification. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3711–3721. PMLR, Nov. 2020.

A. Greenbaum, M. Rozložnik, and Z. Strakoš. Numerical behaviour of the modified gram-schmidt gmres implementation. *BIT Numerical Mathematics*, 37(3):706–719, 1997.

U. Grenander and M. I. Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994.

A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry. AtlasNet: A Papier-M\âch\'e Approach to Learning 3D Surface Generation, July 2018.

P. D. Grünwald. *The minimum description length principle*. MIT press, 2007.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, volume 30, pages 5767–5777. Curran Associates, Inc., 2017.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

K. Gupta. *Stability Quantification of Neural Networks*. PhD thesis, Université Paris-Saclay, 2023.

K. Gupta, F. Kaakai, B. Pesquet-Popescu, and J.-C. Pesquet. Safe design of stable neural networks for fault detection in small uavs. In *International Conference on Computer Safety, Reliability, and Security*, pages 263–275. Springer, 2022a.

K. Gupta, F. Kaakai, B. Pesquet-Popescu, J.-C. Pesquet, and F. D. Malliaros. Multivariate lipschitz analysis of the stability of neural networks. *Frontiers in Signal Processing*, 2:794469, 2022b.

E. Haber and L. Ruthotto. Stable architectures for deep neural networks. *Inverse problems*, 34(1): 014004, 2017.

M. Hallin. Measure transportation and statistical decision theory. *Annual Review of Statistics and Its Application*, 9:401–424, 2022.

M. Hallin, D. Hlubinka, and Š. Hudecová. Fully distribution-free center-outward rank tests for multiple-output regression and manova. *arXiv preprint arXiv:2007.15496*, 2020.

M. Hallin, E. del Barrio, J. Cuesta-Albertos, and C. Matrán. Distribution and quantile functions, ranks and signs in dimension d: A measure transportation approach. *The Annals of Statistics*, 49 (2):1139 – 1165, 2021. doi: 10.1214/20-AOS1996.

M. Hallin, D. Hlubinka, and Š. Hudecová. Efficient fully distribution-free center-outward rank tests for multiple-output regression and manova. *Journal of the American Statistical Association*, 118 (543):1923–1939, 2023.

S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao. ADBench: Anomaly Detection Benchmark. *Neural Information Processing Systems (NeurIPS)*, page 45, 2022.

R. M. Haralick, K. Shanmugam, and I. H. Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, pages 610–621, 1973.

J. Hart. Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. *The Visual Computer*, 12, June 1995. doi: 10.1007/s003710050084.

J. A. Hartigan. Estimation of a Convex Density Contour in Two Dimensions. *Journal of the American Statistical Association*, 82(397):267–270, Mar. 1987. ISSN 0162-1459, 1537-274X. doi: 10.1080/01621459.1987.10478428.

L. Hasenclever, J. M. Tomczak, R. van den Berg, and M. Welling. Variational inference with orthogonal normalizing flows. In *Bayesian Deep Learning, NIPS 2017 workshop*, 2017.

M. H. Hassoun et al. *Fundamentals of artificial neural networks*. MIT press, 1995.

S. Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1998.

J. He, X. Li, D. Yu, H. Zhang, J. Kulkarni, Y. T. Lee, A. Backurs, N. Yu, and J. Bian. Exploring the limits of differentially private deep learning with group-wise clipping. In *The Eleventh International Conference on Learning Representations*, 2022.

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*, pages 770–778, 2016.

K. Helfrich, D. Willmott, and Q. Ye. Orthogonal recurrent neural networks with scaled cayley transform. In *International Conference on Machine Learning*, pages 1969–1978. PMLR, 2018.

D. Hendrycks and K. Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks, Oct. 2018.

D. Hendrycks, M. Mazeika, and T. Dietterich. Deep Anomaly Detection with Outlier Exposure. *arXiv:1812.04606 [cs, stat]*, Jan. 2019.

M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving. *Journal of research of the National Bureau of Standards*, 49(6):409, 1952.

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*, 4(2): 251–257, 1991.

Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960, 2020.

R. L. (https://math.stackexchange.com/users/67071/robert lewis). Solutions to the anticommutator matrix equation. Mathematics Stack Exchange, 2013. URL `https://math.stackexchange.com/q/529461`. URL:https://math.stackexchange.com/q/529461 (version: 2013-10-17).

K. Hu, K. Leino, Z. Wang, and M. Fredrikson. A recipe for improved certifiable robustness: Capacity and data. *arXiv preprint arXiv:2310.02513*, 2023a.

K. Hu, A. Zou, Z. Wang, K. Leino, and M. Fredrikson. Scaling in depth: Unlocking robustness certification on imagenet. *arXiv preprint arXiv:2301.12549*, 2023b.

G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016a.

K. Huang, N. D. Sidiropoulos, and A. P. Liavas. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. *IEEE Transactions on Signal Processing*, 64(19): 5052–5065, 2016b.

L. Huang, X. Liu, B. Lang, A. Yu, Y. Wang, and B. Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

L. Huang, L. Liu, F. Zhu, D. Wan, Z. Yuan, B. Li, and L. Shao. Controllable orthogonalization in training dnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Y. Huang, H. Zhang, Y. Shi, J. Z. Kolter, and A. Anandkumar. Training certifiably robust neural networks with efficient local lipschitz bounds. *Advances in Neural Information Processing Systems*, 34, 2021.

T. Huster, C.-Y. J. Chiang, and R. Chadha. Limitations of the lipschitz constant as a defense against adversarial examples. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 16–29. Springer, 2018.

S. L. Hyland and G. Rätsch. Learning unitary operators with help from u (n). In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

Y. Jia and C. Zhang. Stabilizing gnn for fairness via lipschitz bounds. In *The Second Workshop on New Frontiers in Adversarial Machine Learning*, 2023.

Y. Jiang, D. Krishnan, H. Mobahi, and S. Bengio. Predicting the generalization gap in deep networks with margin distributions. In *International Conference on Learning Representations*, 2019.

C. Jinyin and Y. Dongyong. A study of detector generation algorithms based on artificial immune in intrusion detection system. In *2011 3rd International Conference on Computer Research and Development*, volume 1, pages 4–8. IEEE, 2011.

H. Jooybar, W. W. Fung, M. O'Connor, J. Devietti, and T. M. Aamodt. Gpudet: a deterministic gpu architecture. In *Proceedings of the eighteenth international conference on Architectural support for programming languages and operating systems*, pages 1–12, 2013.

M. Jordan and A. G. Dimakis. Exactly computing the local lipschitz constant of relu networks. *Advances in Neural Information Processing Systems*, 33:7344–7353, 2020.

C. Jung, K. Ligett, S. Neel, A. Roth, S. Sharifi-Malvajerdi, and M. Shenfeld. A new analysis of differential privacy's generalization guarantees. *arXiv preprint arXiv:1909.03577*, 2019.

G. Kaiser and L. H. Hudgins. *A friendly guide to wavelets*, volume 300. Springer, 1994.

W. Karush. Minima of functions of several variables with inequalities as side constraints. *M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago*, 1939.

D. M. Katz, M. J. Bommarito, S. Gao, and P. Arredondo. Gpt-4 passes the bar exam. *Available at SSRN 4389233*, 2023.

K. Kawaguchi, Z. Deng, K. Luh, and J. Huang. Robustness implies generalization via data-dependent generalization bounds. In *International Conference on Machine Learning*, pages 10866–10894. PMLR, 2022.

I. Kerenidis, J. Landman, and N. Mathur. Classical and quantum algorithms for orthogonal neural networks. *arXiv preprint arXiv:2106.07198*, 2021.

G. Khromov and S. P. Singh. Some fundamental aspects about lipschitz continuity of neural network functions. *arXiv preprint arXiv:2302.10886*, 2023.

B. Kiani, R. Balestriero, Y. LeCun, and S. Lloyd. projUNN: efficient method for training deep networks with unitary matrices. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

P. Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

H. Kim, G. Papamakarios, and A. Mnih. The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pages 5562–5571. PMLR, 2021.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

J. Klaus, N. Merk, K. Wiedom, S. Laue, and J. Giesen. Convexity certificates from hessians. *Advances in Neural Information Processing Systems*, 35:6941–6953, 2022.

I. Kobyzev, S. J. Prince, and M. A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11): 3964–3979, 2020.

S. Kolouri, Y. Zou, and G. K. Rohde. Sliced Wasserstein kernels for probability distributions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5258–5267, 2016.

V. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. In *High dimensional probability II*, pages 443–457. Springer, 2000.

A. Korotin, V. Egiazarian, A. Asadulaev, A. Safin, and E. Burnaev. Wasserstein-2 generative networks. In *International Conference on Learning Representations*, 2020.

A. Korotin, L. Li, A. Genevay, J. M. Solomon, A. Filippov, and E. Burnaev. Do neural optimal transport solvers work? a continuous wasserstein-2 benchmark. *Advances in Neural Information Processing Systems*, 34:14593–14605, 2021.

A. Korotin, D. Selikhanovych, and E. Burnaev. Neural optimal transport. In *The Eleventh International Conference on Learning Representations*, 2022.

J. Kovačević, A. Chebira, et al. An introduction to frames. *Foundations and Trends® in Signal Processing*, 2(1):1–94, 2008.

Z. Kovarik. Some iterative methods for improving orthonormality. *SIAM Journal on Numerical Analysis*, 7(3):386–389, 1970. ISSN 00361429.

S. G. Krantz and H. R. Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

H. W. Kuhn and A. W. Tucker. Nonlinear programming proceedings of the second berkeley symposium on mathematical statistics and probability. *Neyman*, pages 481–492, 1951.

R. Kumar, K. Majmundar, D. Nagaraj, and A. S. Suggala. Stochastic re-weighted gradient descent via distributionally robust optimization. *arXiv preprint arXiv:2306.09222*, 2023.

M. J. Kusner, J. Loftus, C. Russell, and R. Silva. Counterfactual fairness. *Advances in neural information processing systems*, 30, 2017.

A. Kuzina, M. Welling, and J. M. Tomczak. Alleviating adversarial attacks on variational autoencoders with mcmc. *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2022.

R. Landauer. Irreversibility and heat generation in the computing process. *IBM journal of research and development*, 5(3):183–191, 1961.

P.-S. Laplace. *Un essai philosophique sur les probabilités*. Imprimerie du cercle social, 1835.

F. Latorre, P. Rolland, and V. Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization. In *International Conference on Learning Representations*, 2019.

S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE transactions on pattern analysis and machine intelligence*, 27(8):1265–1278, 2005.

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 2002.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

J. Lee and D. Kifer. Scaling up differentially private deep learning with fast per-example gradient clipping. *Proceedings on Privacy Enhancing Technologies*, 2021(1), 2021.

J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018a.

K. Lee, H. Lee, K. Lee, and J. Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations*, 2018b.

Y. Lee, C. Finn, and S. Ermon. Relaxing the kolmogorov structure function for realistic computational constraints. In *NeurIPS 2022 Workshop on Information-Theoretic Principles in Cognitive Systems*, 2022.

K. Leino. Limitations of piecewise linearity for efficient robustness certification. *arXiv preprint arXiv:2301.08842*, 2023.

E. León and G. M. Ziegler. Spaces of convex n-partitions. In *New Trends in Intuitive Geometry*, pages 279–306. Springer, 2018.

L. A. Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.

L. A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.

L. A. Levin. The tale of one-way functions. *Problems of Information Transmission*, 39:92–103, 2003.

B. Levy. Graphite v3-1.8.2, 2022. URL `https://github.com/BrunoLevy/GraphiteThree`.

T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of graphics tools*, 8(2):1–15, 2003.

M. Lezcano Casado. Trivializations for gradient-based optimization on manifolds. *Advances in Neural Information Processing Systems*, 32, 2019.

M. Lezcano-Casado and D. Martınez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *International Conference on Machine Learning*, pages 3794–3803. PMLR, 2019.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18 (1):6765–6816, 2017.

M. Li, P. Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008.

Q. Li, S. Haque, C. Anil, J. Lucas, R. B. Grosse, and J.-H. Jacobsen. Preventing gradient attenuation in lipschitz constrained convolutional networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, Cambridge, MA, 2019a. MIT Press.

S. Li, K. Jia, Y. Wen, T. Liu, and D. Tao. Orthogonal deep neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(4):1352–1368, 2019b.

X. Li, F. Tramer, P. Liang, and T. Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*, 2021.

R. R. Lin, H. Z. Zhang, and J. Zhang. On reproducing kernel banach spaces: Generic definitions and unified framework of constructions. *Acta Mathematica Sinica, English Series*, 38(8):1459–1483, 2022.

D. Linsley, S. Eberhardt, T. Sharma, P. Gupta, and T. Serre. What are the visual features underlying human versus machine vision? In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2706–2714, 2017.

Z. Lipton, J. McAuley, and A. Chouldechova. Does mitigating ml's impact disparity require treatment disparity? *Advances in neural information processing systems*, 31, 2018.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

X. Liu, L. Xie, Y. Wang, J. Zou, J. Xiong, Z. Ying, and A. V. Vasilakos. Privacy and security issues in deep learning: A survey. *IEEE Access*, 9:4566–4593, 2020.

S.-Y. Lo, P. Oza, and V. M. Patel. Adversarially robust one-class novelty detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

N. Loizou, S. Vaswani, I. H. Laradji, and S. Lacoste-Julien. Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence. In *International Conference on Artificial Intelligence and Statistics*, pages 1306–1314. PMLR, 2021.

J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International conference on artificial intelligence and statistics*, pages 1540–1552. PMLR, 2020.

Y. Lu, A. Zhong, Q. Li, and B. Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, pages 3276–3285. PMLR, 2018.

B. Ma, J. Zhou, Y.-S. Liu, and Z. Han. Towards better gradient consistency for neural signed distance functions via level set alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17724–17734, 2023.

E. Macías-Virgós, M. J. Pereira-Sáez, and D. Tanré. Cayley transform on stiefel manifolds. *Journal of Geometry and Physics*, 123:53–60, 2018.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

S. Maneewongvatana and D. M. Mount. Analysis of approximate nearest neighbor searching with clustered point sets. In *ALENEX 99*, 1999.

R. Mangal, K. Leino, Z. Wang, K. Hu, W. Yu, C. Pasareanu, A. Datta, and M. Fredrikson. Is certifying lp robustness still worthwhile? *arXiv preprint arXiv:2310.09361*, 2023.

P. Manupriya, R. K. Das, S. Biswas, S. Chandhok, and S. N. Jagarlapudi. Empirical optimal transport between conditional distributions. *arXiv preprint arXiv:2305.15901*, 2023.

V. A. Marchenko and L. A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Matematicheskii Sbornik*, 114(4):507–536, 1967.

P. S. Mashhadi, S. Nowaczyk, and S. Pashami. Parallel orthogonal deep neural network. *Neural Networks*, 140:167–183, 2021.

A. Mathiasen, F. Hvilshøj, J. R. Jørgensen, A. Nasery, and D. Mottin. Faster orthogonal parameterization with householder matrices. In *ICML, Workshop Proceedings*, 2020.

D. A. McAllester. Some pac-bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 230–234, 1998.

H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

D. J. Meagher. *Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer*. Electrical and Systems Engineering Department Rensseiaer Polytechnic ..., 1980.

N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35, 2021.

D. Meunier, M. Pontil, and C. Ciliberto. Distribution regression with sliced Wasserstein kernels. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15501–15523. PMLR, 17–23 Jul 2022a.

L. Meunier, B. J. Delattre, A. Araujo, and A. Allauzen. A dynamical system perspective for lipschitz neural networks. In *International Conference on Machine Learning*, pages 15484–15500. PMLR, 2022b.

Z. Mhammedi, A. Hellicar, A. Rahman, and J. Bailey. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *International Conference on Machine Learning*, pages 2401–2409. PMLR, 2017.

C. A. Micchelli and M. Pontil. A function representation for learning in banach spaces. In *International Conference on Computational Learning Theory*, pages 255–269. Springer, 2004.

I. Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE, 2017.

I. Mironov, K. Talwar, and L. Zhang. R\'enyi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.

R. Mises and H. Pollaczek-Geiringer. Praktische verfahren der gleichungsauflösung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 9 (1):58–77, 1929.

T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021 (12):124003, 2021.

K. Nar, O. Ocal, S. S. Sastry, and K. Ramchandran. Cross-entropy loss and low-rank features have responsibility for adversarial examples. *arXiv preprint arXiv:1901.08360*, 2019.

R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Y. E. Nesterov. A method for solving the convex programming problem with convergence rate o(1/(k*k)). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983.

B. Neyshabur, S. Bhojanapalli, and N. Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018.

V. Niculae. Optimizing with constraints: reparametrization and geometry. `https://vene.ro/blog/mirror-descent.html`, 2020. Accessed: 2023-10-02.

V. Niculae and M. Blondel. A regularized framework for sparse and structured neural attention. *Advances in neural information processing systems*, 30, 2017.

E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. *Advances in Neural Information Processing Systems*, 32, 2019.

E. Nijkamp, M. Hill, T. Han, S.-C. Zhu, and Y. N. Wu. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5272–5280, 2020.

J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 1999.

T. Novello, G. Schardong, L. Schirmer, V. da Silva, H. Lopes, and L. Velho. Exploring Differential Geometry in Neural Implicits, Aug. 2022.

C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

J. Oliva, W. Neiswanger, B. Póczos, J. Schneider, and E. Xing. Fast distribution to real regression. In *International Conference on Artificial Intelligence and Statistics*, 2014.

T. Pang, K. Xu, Y. Dong, C. Du, N. Chen, and J. Zhu. Rethinking softmax cross-entropy loss for adversarial robustness. In *International Conference on Learning Representations*, 2019.

H. Papadopoulos. Guaranteed coverage prediction intervals with gaussian process regression. *arXiv preprint arXiv:2310.15641*, 2023.

N. Papernot and T. Steinke. Hyperparameter tuning with renyi differential privacy. In *International Conference on Learning Representations*, 2022.

N. Papernot, A. Thakurta, S. Song, S. Chien, and Ú. Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9312–9321, 2021.

G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

P. Pauli, A. Havens, A. Araujo, S. Garg, F. Khorrami, F. Allgöwer, and B. Hu. Semidefinite programs for computing lipschitz bounds of neural networks with maxmin activations. In *preprint*, 2023.

B. Pearlmutter and R. Rosenfeld. Chaitin-kolmogorov complexity and generalization in neural networks. *Advances in neural information processing systems*, 3, 1990.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

K. Pei, D. Bieber, K. Shi, C. Sutton, and P. Yin. Can large language models reason about program invariants? In *International Conference on Machine Learning*, pages 27496–27520. PMLR, 2023.

G. Peyré, M. Cuturi, et al. Computational optimal transport. *Center for Research in Economics and Statistics Working Papers*, 2017.

T. Pinder and D. Dodd. Gpjax: A Gaussian process framework in JAX. *Journal of Open Source Software*, 7(75):4455, 2022. doi: 10.21105/joss.04455. URL https://doi.org/10.21105/joss.04455.

A. Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

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

N. Ponomareva, H. Hazimeh, A. Kurakin, Z. Xu, C. Denison, H. B. McMahan, S. Vassilvitskii, S. Chien, and A. Thakurta. How to dp-fy ml: A practical guide to machine learning with differential privacy. *arXiv preprint arXiv:2303.00654*, 2023.

A. Pouly. *Continuous models of computation: from computability to complexity*. PhD thesis, PhD thesis, Ecole Polytechnique and Unidersidade Do Algarve, 2015.

M. Pourreza, B. Mohammadi, M. Khaki, S. Bouindour, H. Snoussi, and M. Sabokrou. G2D: Generate to Detect Anomaly. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2002–2011, Waikoloa, HI, USA, Jan. 2021. IEEE. doi: 10.1109/WACV48630.2021.00205.

B. Prach and C. H. Lampert. Almost-orthogonal layers for efficient general-purpose lipschitz networks. In *European Conference on Computer Vision*, pages 350–365. Springer, 2022.

B. Prach, F. Brau, G. Buttazzo, and C. H. Lampert. 1-lipschitz layers compared: Memory, speed, and certifiable robustness, 2023.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

X. Qi, J. Wang, Y. Chen, Y. Shi, and L. Zhang. Lipsformer: Introducing lipschitz continuity to vision transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

H. Qian and M. N. Wegman. L2-nonexpansive neural networks. In *International Conference on Learning Representations*, 2018.

E. Rabkin. Full investigation of the matrix equation ax+xb=c and specifically of the equation ax-xa=c. *St. Petersburg Mathematical Journal*, 26(1):117–130, 2015.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, 2007.

C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5(53):2607, 2020. doi: 10.21105/joss.02607. URL https://doi.org/10.21105/joss.02607.

S. Rayana. ODDS library, 2016. URL http://odds.cs.stonybrook.edu.

J. Richter-Powell, J. Lorraine, and B. Amos. Input convex gradient networks. *arXiv preprint arXiv:2111.12187*, 2021.

J. Richter-Powell, Y. Lipman, and R. T. Chen. Neural conservation laws: A divergence-free perspective. *Advances in Neural Information Processing Systems*, 35:38075–38088, 2022.

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

O. Rojo and H. Rojo. Some results on symmetric circulant matrices and on symmetric centrosymmetric matrices. *Linear algebra and its applications*, 392:211–233, 2004.

F. Rosenblatt. The perceptron - a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York, January 1957.

A. Ross and F. Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

V. Roulet and Z. Harchaoui. An elementary approach to convergence guarantees of optimization algorithms for deep networks. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 84–91. IEEE, 2019.

V. Roulet and Z. Harchaoui. On the smoothing of deep networks. In *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2021.

V. Roulet, S. Srinivasa, M. Fazel, and Z. Harchaoui. Iterative linear quadratic optimization for nonlinear control: Differentiable programming algorithmic templates. *arXiv preprint arXiv:2207.06362*, 2022.

M. Rousson and N. Paragios. Shape priors for level set representations. In *European Conference on Computer Vision*, pages 78–92. Springer, 2002.

M. Rudelson and R. Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 54(4):21–es, 2007.

M. Rudelson and R. Vershynin. Non-asymptotic theory of random matrices: extreme singular values. In *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures*, pages 1576–1602. World Scientific, 2010.

L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018.

Y. Saad and M. H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3379–3388, 2018.

T. W. Sager. An Iterative Method for Estimating a Multivariate Mode and Isopleth. *Journal of the American Statistical Association*, 74(366):329, June 1979. ISSN 01621459. doi: 10.2307/2286331.

T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016.

A. Salmona, V. De Bortoli, J. Delon, and A. Desolneux. Can push-forward generative models fit multimodal distributions? *Advances in Neural Information Processing Systems*, 35:10766–10779, 2022.

T. Sander, P. Stock, and A. Sablayrolles. Tan without a burn: Scaling laws of dp-sgd. *arXiv preprint arXiv:2210.03403*, 2022.

A. Sanyal, P. H. Torr, and P. K. Dokania. Stable rank normalization for improved generalization in neural networks and gans. In *International Conference on Learning Representations*, 2020.

K. Scaman and A. Virmaux. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 3839–3848, 2018.

R. Schaeffer, M. Khona, Z. Robertson, A. Boopathy, K. Pistunova, J. W. Rocks, I. R. Fiete, and O. Koyejo. Double descent demystified: Identifying, interpreting & ablating the sources of a deep learning puzzle. *arXiv preprint arXiv:2303.14151*, 2023.

J. Schmidhuber. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Networks*, 10(5):857–873, 1997.

J. Schmidhuber. Algorithmic theories of everything. *arXiv preprint quant-ph/0011122*, 2000.

B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support Vector Method for Novelty Detection. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001a.

B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001b.

M. Schubiger, G. Banjac, and J. Lygeros. GPU acceleration of ADMM for large-scale quadratic programming. *Journal of Parallel and Distributed Computing*, 144:55–67, 2020. doi: 10.1016/j.jpdc.2020.05.021.

D. Schuurmans. Memory augmented large language models are computationally universal. *arXiv preprint arXiv:2301.04589*, 2023.

D. Scieur, G. Gidel, Q. Bertrand, and F. Pedregosa. The curse of unrolling: Rate of differentiating through optimization. *Advances in Neural Information Processing Systems*, 35:17133–17145, 2022.

M. Serrurier, F. Mamalet, A. González-Sanz, T. Boissin, J.-M. Loubes, and E. Del Barrio. Achieving robustness in classification using optimal transport with hinge regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 505–514, 2021.

M. Serrurier, F. Mamalet, T. Fel, L. Béthune, and T. Boissin. On the explainable properties of 1-lipschitz neural networks: An optimal transport perspective. In *Advances in Neural Information Processing Systems*, 2023.

N. Sharp and A. Jacobson. Spelunking the deep: Guaranteed queries on general neural implicit surfaces via range analysis. *ACM Transactions on Graphics*, 41(4):1–16, July 2022. ISSN 0730-0301, 1557-7368. doi: 10.1145/3528223.3530155.

Y. Shavit and B. Gjura. Exploring the use of lipschitz neural networks for automating the design of differentially private mechanisms. Technical report, Technical Report, 2019.

J. Shawe-Taylor and R. C. Williamson. A pac analysis of a bayesian estimator. In *Proceedings of the tenth annual conference on Computational learning theory*, pages 2–9, 1997.

L. Simon et al. *Lectures on geometric measure theory*. The Australian National University, Mathematical Sciences Institute, Centre . . . , 1983.

U. Simsekli, O. Sener, G. Deligiannidis, and M. A. Erdogdu. Hausdorff dimension, heavy tails, and generalization in neural networks. *Advances in Neural Information Processing Systems*, 33: 5138–5151, 2020.

G. Singh, T. Gehr, M. Püschel, and M. Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019.

S. Singla and S. Feizi. Fantastic four: Differentiable bounds on singular values of convolution layers. In *International Conference on Learning Representations (ICLR)*, 2021a.

S. Singla and S. Feizi. Skew orthogonal convolutions. In *International Conference on Machine Learning*, pages 9756–9766. PMLR, 2021b.

S. Singla and S. Feizi. Improved techniques for deterministic l2 robustness. *Advances in Neural Information Processing Systems*, 35:16110–16124, 2022.

S. Singla, S. Singla, and S. Feizi. Improved deterministic l2 robustness on cifar-10 and cifar-100. In *International Conference on Learning Representations*, 2021.

J. Sipple. Interpretable, Multidimensional, Multimodal Anomaly Detection with Negative Sampling for Detection of Device Failure. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9016–9025. PMLR, Nov. 2020.

M. Slater. Lagrange multipliers revisited. In *Traces and emergence of nonlinear programming*, pages 293–306. Springer, 2013.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

I. Sobel, G. Feldman, et al. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

L. Song, R. Shokri, and P. Mittal. Privacy risks of securing machine learning models against adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 241–257, 2019.

X. Song, J. Duan, W. Wang, S. E. Li, C. Chen, B. Cheng, B. Zhang, J. Wei, and X. S. Wang. LipsNet: A smooth and robust neural network with adaptive Lipschitz constant for high accuracy optimal control. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*. PMLR, 23–29 Jul 2023.

A. Srinivasan and E. Todorov. Graphical newton. *arXiv preprint arXiv:1508.00952*, 2015.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.

B. Stasiak and M. Yatsymirskyy. Fast orthogonal neural networks. In *International Conference on Artificial Intelligence and Soft Computing*, pages 142–149. Springer, 2006.

B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020. doi: 10.1007/s12532-020-00179-2.

T. Stibor, P. Mohr, J. Timmis, and C. Eckert. Is negative selection appropriate for anomaly detection? In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, GECCO '05, pages 321–328, New York, NY, USA, June 2005. Association for Computing Machinery. doi: 10.1145/1068009.1068061.

M. H. Stone. Applications of the theory of boolean rings to general topology. *Transactions of the American Mathematical Society*, 41(3):375–481, 1937.

S. J. Szarek. Metric entropy of homogeneous spaces. *Banach Center Publications*, 43(1):395–410, 1998.

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

U. Tanielian and G. Biau. Approximating lipschitz continuous functions with groupsort neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 442–450. PMLR, 2021.

Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003.

W. Thomson. 9. the kinetic theory of the dissipation of energy. *Proceedings of the Royal Society of Edinburgh*, 8:325–334, 1875.

V. Tjeng, K. Y. Xiao, and R. Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019.

I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

R. Tomsett, A. Widdicombe, T. Xing, S. Chakraborty, S. Julier, P. Gurram, R. Rao, and M. Srivastava. Why the failure? how adversarial examples can provide insights for interpretable machine learning. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 838–845. IEEE, 2018.

F. Tramer and D. Boneh. Differentially private learning needs better features (or much more data), 2021.

L. N. Trefethen and D. Bau. *Numerical linear algebra*, volume 181. Siam, 2022.

R. M. Trigub and E. S. Belinsky. *Fourier analysis and approximation of functions*. Springer Science & Business Media, 2004.

A. Trockman and J. Z. Kolter. Orthogonalizing convolutional layers with the cayley transform. In *International Conference on Learning Representations*, 2021.

Y. Tsuzuku, I. Sato, and M. Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 6541–6550. Curran Associates, Inc., 2018.

F. Uhlig. Constructive ways for generating (generalized) real orthogonal matrices as products of (generalized) symmetries. *Linear Algebra and its Applications*, 332:459–467, 2001.

T. Uscidda and M. Cuturi. The monge gap: A regularizer to learn all transport maps. In *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2023.

D. Usynin, A. Ziller, M. Knolle, A. Trask, K. Prakash, D. Rueckert, and G. Kaissis. An automatic differentiation system for the age of differential privacy. *arXiv preprint arXiv:2109.10573*, 2021.

L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

H. A. Van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, 13(2): 631–644, 1992.

S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu. Scikit-image: image processing in Python. *PeerJ*, 2:e453, 2014.

V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

V. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

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. *Advances in neural information processing systems*, 32, 2019.

N. Vazou. *Liquid Haskell: Haskell as a theorem prover*. University of California, San Diego, 2016.

P. Viallard, E. G. VIDOT, A. Habrard, and E. Morvant. A pac-bayes analysis of adversarial robustness. *Advances in Neural Information Processing Systems*, 34:14421–14433, 2021.

C. Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

A. Virmaux and K. Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

U. von Luxburg and O. Bousquet. Distance-based classification with lipschitz functions. *J. Mach. Learn. Res.*, 5:669–695, 2004.

J. Von Neumann. First draft of a report on the edvac. *internal report*, 1945.

H. F. Walker and P. Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

J. Wang, Y. Chen, R. Chakraborty, and S. X. Yu. Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11505–11515, 2020.

R. Wang and I. Manchester. Direct parameterization of lipschitz-bounded deep networks. In *International Conference on Machine Learning*, pages 36093–36110. PMLR, 2023.

S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, and J. Z. Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems*, 34:29909–29921, 2021.

W. Wang, Z. Dang, Y. Hu, P. Fua, and M. Salzmann. Backpropagation-friendly eigendecomposition. *Advances in Neural Information Processing Systems*, 32, 2019a.

X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2022a.

Y.-X. Wang, J. Lei, and S. E. Fienberg. Learning with differential privacy: Stability, learnability and the sufficiency and necessity of erm principle. *The Journal of Machine Learning Research*, 17 (1):6353–6392, 2016.

Y.-X. Wang, B. Balle, and S. P. Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1226–1235. PMLR, 2019b.

Z. Wang, G. Prakriya, and S. Jha. A quantitative geometric approach to neural-network smoothness. *Advances in Neural Information Processing Systems*, 35:34201–34215, 2022b.

X. Warin. The groupmax neural network approximation of convex functions. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

T. Weber, N. Heess, L. Buesing, and D. Silver. Credit assignment techniques in stochastic computation graphs. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2650–2660. PMLR, 2019.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pages 5276–5285. PMLR, 2018a.

T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. In *International Conference on Learning Representations*, 2018b.

D. V. Widder. *Advanced calculus*. Courier Corporation, 2012.

J. Wu, A. A. Ghomi, D. Glukhov, J. C. Cresswell, F. Boenisch, and N. Papernot. Augment then smooth: Reconciling differential privacy with certified robustness. *arXiv preprint arXiv:2306.08656*, 2023.

Y. Wu and K. He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

N. Xiao and X. Liu. Solving optimization problems over the stiefel manifold by smooth exact penalty function. *arXiv preprint arXiv:2110.08986*, 2021.

A. Xu and M. Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. *Advances in Neural Information Processing Systems*, 30, 2017.

K. Xu, Z. Shi, H. Zhang, Y. Wang, K.-W. Chang, M. Huang, B. Kailkhura, X. Lin, and C.-J. Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33:1129–1141, 2020.

X. Xu, L. Li, Y. Cheng, S. Mukherjee, A. H. Awadallah, and B. Li. Certifiably robust transformers with 1-lipschitz self-attention. In *preprint*, 2022a.

X. Xu, L. Li, and B. Li. Lot: Layer-wise orthogonal training on improving l2 certified robustness. *arXiv preprint arXiv:2210.11620*, 2022b.

Y. Xu, S. Zhao, J. Song, R. Stewart, and S. Ermon. A theory of usable information under computational constraints. In *International Conference on Learning Representations*, 2019.

R. Yamauchi, J. Sakurai, R. Furukawa, and T. Matsubayashi. Optimizing implicit neural representations from point clouds via energy-based models. *arXiv preprint arXiv:2311.02601*, 2023.

H. Yang, Y. Sun, G. Sundaramoorthi, and A. Yezzi. Steik: Stabilizing the optimization of neural signed distance functions and finer shape representation. *Advances in neural information processing systems*, 2023.

X. Yang, H. Zhang, W. Chen, and T.-Y. Liu. Normalized/clipped sgd with perturbation for differentially private non-convex optimization. *arXiv preprint arXiv:2206.13033*, 2022.

Y.-Y. Yang, C. Rashtchian, H. Zhang, R. R. Salakhutdinov, and K. Chaudhuri. A closer look at accuracy vs. robustness. *Advances in Neural Information Processing Systems*, 33, 2020.

D. Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

Y. Yoshida and T. Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, G. Cormode, and I. Mironov. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*, 2021.

H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar. Adversarially learned anomaly detection. In *2018 IEEE International conference on data mining (ICDM)*, pages 727–736. IEEE, 2018.

S. Zhai, T. Likhomanenko, E. Littwin, D. Busbridge, J. Ramapuram, Y. Zhang, J. Gu, and J. M. Susskind. Stabilizing transformer training by preventing attention entropy collapse. In *International Conference on Machine Learning*, pages 40770–40803. PMLR, 2023.

B. Zhang, T. Cai, Z. Lu, D. He, and L. Wang. Towards certifying l-infinity robustness using neural networks with l-inf-dist neurons. In *International Conference on Machine Learning*, pages 12368–12379. PMLR, 2021a.

B. Zhang, D. Jiang, D. He, and L. Wang. Rethinking lipschitz neural networks for certified l-infinity robustness. *arXiv preprint arXiv:2210.01787*, 2022.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021b.

H. Zhang, Y. Xu, and J. Zhang. Reproducing kernel banach spaces for machine learning. *Journal of Machine Learning Research*, 10(12), 2009.

H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018a.

H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

J. Zhang, Q. Lei, and I. Dhillon. Stabilizing gradients for deep neural networks via efficient svd parameterization. In *International Conference on Machine Learning*, pages 5806–5814. PMLR, 2018b.

Z. Zhou, J. Liang, Y. Song, L. Yu, H. Wang, W. Zhang, Y. Yu, and Z. Zhang. Lipschitz generative adversarial nets. In *International Conference on Machine Learning*, pages 7584–7593. PMLR, 2019.

Y. Zhu and Y.-X. Wang. Possion subsampled rényi differential privacy. In *International Conference on Machine Learning*, pages 7634–7642. PMLR, 2019.

Y. Zhu and Y.-X. Wang. Improving sparse vector technique with renyi differential privacy. *Advances in Neural Information Processing Systems*, 33:20249–20258, 2020.

A. Ziller, D. Usynin, M. Knolle, K. Prakash, A. Trask, R. Braren, M. Makowski, D. Rueckert, and G. Kaissis. Sensitivity analysis in differentially private machine learning using hybrid automatic differentiation. *arXiv preprint arXiv:2107.04265*, 2021.

Appendix A

Optimization as a layer

A.1 Differentiable invertible transformation on images coordinates

The content of this section is unpublished.

Chain rule can be applied to a lot of transformations - including the ones arising in data augmentation itself. This allows us to compute the derivative of the loss w.r.t hyper-parameters of the data augmentation itself. Below, we illustrate this approach on 2D images, and more specifically on the "angle" parameter of rotations. Let $\hat{f} : [|0, n-1|]^2 \to [0,1]$ be the original image that assigns luminosity to each integer coordinates (i, j). Equivalently, $\hat{f}$ can be seen as an element of $[0,1]^{n\times n}$

From discrete domain images to infinite continuous world

We define $f : \mathbb{R}^2 \to [0,1]$ as the centered interpolation of $\hat{f}$ on the whole space, such that $f(-1,-1) = \hat{f}(0,0)$ and $f(1,1) = \hat{f}(n-1, n-1)$, i.e for integer coordinates (i, j) we have $x(i) = \frac{2i-(n-1)}{n-1}$ and $y(j) = \frac{2j-(n-1)}{n-1}$. And for continuous coordinates x, y we have $i(x) = \lfloor g(x) \rceil$ and $j(y) = \lfloor g(y) \rceil$ we have:

$$g(x) = \frac{n-1}{2}(x+1).$$

The function g assigns to each continuous pair of coordinates (x, y) the value of the nearest pixel.

Exemple A.1. Zero padding.

With zero padding, we chose $f(x, y) = 0$ whenever $\|(x, y)\|_\infty \geq 1$. We introduce the notations $\overline{i}(x) = \lceil g(x) \rceil$, $\underline{i}(x) = \lfloor g(x) \rfloor$, $\overline{j}(y) = \lceil g(y) \rceil$ and $\underline{j}(y) = \lfloor g(y) \rfloor$ for clarity. Then for linear interpolation we have:

$$f(x,y) = \begin{bmatrix} \overline{i}(x) - g(x) & g(x) - \underline{i}(x) \end{bmatrix} \begin{bmatrix} \hat{f}(\underline{i}(x), \underline{j}(y)) & \hat{f}(\underline{i}(x), \overline{j}(y)) \\ \hat{f}(\overline{i}(x), \underline{j}(y)) & \hat{f}(\overline{i}(x), \overline{j}(y)) \end{bmatrix} \begin{bmatrix} \overline{j}(y) - g(y) \\ g(y) - \underline{j}(y) \end{bmatrix}.$$

Other cases of padding (such as nearest) or interpolation (such as cubic) can be handled similarly but it can be a bit cumbersome to write it down here.

Invertible transformation in $\mathbb{R}^2$

Assume that the transformed image $s : \mathbb{R}^2 \to [0,1]$ is given by:

$$s(x, y) = f(M_\theta^{-1}(x, y)),$$

where $M_\theta : \mathbb{R}^2 \to \mathbb{R}^2$ is an invertible mapping parametrized by $\theta \in \mathbb{R}^d$, that characterizes the transformation of pixel coordinates.

Exemple A.2. Isometries.

The case of isometries (combination of rotation of angle θ_1 and translation $[\theta_2, \theta_3]$) can be handled a follow:

$$M_\theta(x, y) = \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{pmatrix} + \begin{pmatrix} \theta_2 & \theta_3 \end{pmatrix}.$$

Note that here we have:

$$\begin{aligned} M_\theta^{-1}(x, y) &= \left(\begin{pmatrix} x & y \end{pmatrix} - \begin{pmatrix} \theta_2 & \theta_3 \end{pmatrix}\right) \begin{pmatrix} \cos -\theta_1 & \sin -\theta_1 \\ -\sin -\theta_1 & \cos -\theta_1 \end{pmatrix} \\ &= \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{pmatrix} - \begin{pmatrix} \theta_2\cos\theta_1 - \theta_3\sin\theta_1 & -\theta_2\sin\theta_1 + \theta_3\cos\theta_1 \end{pmatrix} \end{aligned} \tag{A.1}$$

Notice that the inverse mapping M^{-1} is itself an isometry. This will be of particular interest in the following sections.

Recover discrete image from continuous world

The final image $\hat{s} : [|0, n-1|]^2 \to [0, 1]$ is defined as $\hat{s}(i, j) = s(x(i), y(j))$. Once again the function $\hat{s}$ is better understood as an element of $[0, 1]^{n \times n}$ which verifies $\hat{s}_{i,j} = \hat{s}(i, j) = s(x(i), y(j))$.

Gradient of the loss $\mathcal{L}$ with respect to parameters θ

We are interested in the Jacobian $J_\theta \hat{s} \in \mathbb{R}^{(n^2) \times d}$ since $\hat{s} \in \mathbb{R}^{n \times n}$ and $\theta \in \mathbb{R}^d$. Note that this Jacobian can be estimated either row-by-row (backward autodiff) by computing $\nabla_\theta \hat{s}_{i,j} \in \mathbb{R}^d$ for all (i, j), either column-by-column (forward autodiff) by computing $\frac{\partial \hat{s}}{\partial \theta_k} \in \mathbb{R}^{n^2}$ for all $1 \leq k \leq d$.

Derivative of continuous s with respect to transformation parameter θ

We focus on the former case since back-propagation is the most popular algorithm. In this case:

$$\nabla_\theta s(x, y) = \nabla_{(x,y)} f(M_\theta^{-1}(x, y)) \left(J_\theta M_\theta^{-1}(x, y)\right).$$

Note that $J_\theta M_\theta^{-1}(x, y) \in \mathbb{R}^{2 \times d}$ where the Jacobian of $M_\theta^{-1}(x, y)$ is taken with respect to vector θ while the variables (x, y) are held constant. Note that $\nabla_{(x,y)} f(M_\theta^{-1}(x, y)) \in \mathbb{R}^2$ corresponds to image derivative evaluated in (x, y) in continuous domain. It can be computed from the interpolation directly (which is derivable almost everywhere with respect to x and y). Or it can be estimated in discrete domain $\hat{f}(i(\hat{x}), j(\hat{y}))$ by applying Sobel filter in both directions on coordinates:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = M_\theta^{-1}(x, y).$$

Interestingly, in the case of linear interpolation for f, the image derivative $\nabla_{(x,y)} s$ end up being equal to the discrete gradient computed by Sobel filter on s (proof left as an exercise for reader).

Derivative of discrete $\hat{s}$ with respect to transformation parameter θ

The final expression for $\hat{s}_{i,j}$ yields:

$$\nabla_\theta \hat{s}_{i,j} = \nabla_{(x,y)} f(M_\theta^{-1}(x(i), y(j))) \left(J_\theta M_\theta^{-1}(x(i), y(j))\right).$$

Applied to the whole vector $\hat{s}$ we get:

$$J_\theta \hat{s} = \left[\nabla_{(x,y)} f(M_\theta^{-1}(x(i), y(j))) \left(J_\theta M_\theta^{-1}(x(i), y(j))\right)\right]_{0 \leq i,j \leq n-1}.$$

The notation $[\cdot]_{0 \leq i,j \leq n-1}$ creates a vector of shape $n^2 \times S$ from a set of n^2 vectors $\cdot$ of shape S, indexed by pairs (i, j). The reader can check that $J_\theta \hat{s}$ has the expected dimension $n^2 \times d$. However computing it this way would be rather inefficient.

Vector Jacobian product (VJP) for efficient back-propagation

To leverage the advantage of backpropagation we take a step back to the problem of interest: computing $\nabla_\theta \mathcal{L}(\theta)$. Examples of such loss $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ are given in the next section.

The chain rule gives:

$$\nabla_\theta \mathcal{L}(\theta) = \nabla_{\hat{s}} \mathcal{L}(\hat{s}) J_\theta \hat{s}.$$

The computation of $v = \nabla_{\hat{s}} \mathcal{L}(\hat{s}) \in \mathbb{R}^{n^2}$ (the derivative of the loss with respect to transformed image) is usually handled by backward Autodiff transparently. Then, we factorize the VJP in the following way:

$$v J_\theta \hat{s} = v(J_{x,y} \left[s(x(i), y(j))\right]_{0 \leq i,j \leq n-1}) \odot (J_\theta \left[M_\theta^{-1}(x(i), y(j))\right]_{0 \leq i,j \leq n-1}). \tag{A.2}$$

Remark A.1. Interpretations of terms.

Observe that $J_{x,y} \left[s(x(i), y(j))\right]_{0 \leq i,j \leq n-1} \in \mathbb{R}^{n^2 \times 2}$ is the spatial derivative of the transformed image s in continuous domain. The first (resp. second) "column" (actually a 2D array if the image is unflatten) of the Jacobian corresponds to spatial derivative $D_x \in \mathbb{R}^{n^2}$ (resp. $D_y \in \mathbb{R}^{n^2}$) taken in direction x (resp. y), as computed by Sobel filters. For clarity we will write $[D_x, D_y] = J_{x,y} \left[s(x(i), y(j))\right]_{0 \leq i,j \leq n-1}$. Note that $J_\theta \left[M_\theta^{-1}(x(i), y(j))\right]_{0 \leq i,j \leq n-1} \in \mathbb{R}^{n^2 \times 2 \times d}$ are the derivatives of the $n^2 \times 2$ continuous coordinates with respect to θ. Here $\odot$ denotes a batched vector-matrix product, where the batch dimension is of size n^2 and involves n^2 simultaneous vector-matrix products between vectors of size 2 and matrices of size $2 \times d$.

A.1.1 VJP for invertible affine transformations

We now focus on the case where $M_\theta : \mathbb{R}^2 \to \mathbb{R}^2$ is an affine mapping. In this case it well known that if $M_\theta(u) = uL_\theta + B_\theta$ for matrix $L_\theta \in \mathbb{R}^{2 \times 2}$ and vector $B_\theta \in \mathbb{R}^2$, then the inverse function $M_\theta^{-1}(u) = uL_\theta^{-1} - B_\theta L_\theta^{-1}$ is itself an affine mapping. Let $[X, Y] = [x(i), y(j)]_{1 \leq i,j \leq n-1} \in \mathbb{R}^{n^2 \times 2}$ be the $n^2 \times 2$ matrix of continuous coordinates. Then we have:

$$J_\theta \left[M_\theta^{-1}(x(i), y(j))\right]_{0 \leq i,j \leq n-1} = J_\theta([X, Y]L_\theta^{-1} - 1_{n^2} \otimes B_\theta L_\theta^{-1}).$$

The operation $[X, Y]L_\theta^{-1} \in \mathbb{R}^{n^2 \times 2}$ is a vectorized matrix-vector product, which end up being formulated as a matrix-matrix product. Broadcasting is used on $L_\theta^{-1} B_\theta \in \mathbb{R}^2$ using outer product $\otimes$ with $1_{n^2} \in \mathbb{R}^{n^2 \times 1}$ to ensure the difference is taken on objects of the same shape.

VJP for rotations

In the case of rotation matrices without translation we have $B_\theta = 0$ and $L_\theta^{-1} = L_\theta^T$ so the above formula simplifies in:

$$J_\theta \left[M_\theta^{-1}(x(i), y(j))\right]_{0 \leq i,j \leq n-1} = J_\theta([X, Y]L_\theta^T).$$

In this case this simplifies to:

$$J_\theta([X, Y]L_\theta^T) = [X, Y](J_\theta L_\theta^T).$$

Notice that $(J_\theta L_\theta^T) \in \mathbb{R}^{2\times 2\times d}$ so the product with $[X, Y] \in \mathbb{R}^{n^2\times 2}$ is well defined and yields $[X, Y](J_\theta L_\theta^T) \in \mathbb{R}^{n^2\times 2\times d}$ as expected. As a rotation is characterized by a single angle we have $d = 1$ which allows further simplifications.

Finally we get:

$$\nabla_\theta \mathcal{L}(\theta) = v J_\theta \hat{s} = v \left([D_x, D_y] \odot \left([X, Y](J_\theta L_\theta^T)\right)\right), \tag{A.3}$$

where:

- $\nabla_\theta \mathcal{L}(\theta) \in \mathbb{R}^d$ is the *tangent* vector of interest, that we want to compute with a query to 'tape.gradient'.
- $v = \nabla_{\hat{s}} \mathcal{L}(\hat{s}) \in \mathbb{R}^{n^2}$ is a *cotangent* vector, corresponding to the 'dy' (or 'upstream)' variable in 'tf.custom_gradient' decorated functions. This is the gradient of the loss with respect to the transformed image.
- $[D_x, D_y] \in \mathbb{R}^{n^2\times 2}$ are the spatial derivatives obtained by applying Sobel filters G_x, G_y on the transformed image $\hat{s}$. We have $D_x = G_x \circledast \hat{s}$ and $D_y = G_y \circledast \hat{s}$ where $\circledast$ denotes the convolution operation.
- $[X, Y] \in [-1, 1]^{n^2\times 2}$ is the set of continuous coordinates obtained from integer indexes $1 \leq i, j \leq n - 1$. This vector is constant during training.
- $(J_\theta L_\theta^T) \in \mathbb{R}^{2\times 2\times d}$ is the Jacobian of rotation matrix. Since $d = 1$ it has the same size as the matrix itself, but it is not a rotation matrix in general.
- the special structure of the problem allows to use batched vector-matrix products $\odot$ between $[D_x, D_y]$ and $[X, Y](J_\theta L_\theta^T)$, which yields a matrix of shape $n^2 \times d$.

Exemple A.3. Batch of images and multiple channels.

We consider the case where we apply the transformation simultaneously on b images having c channels each (in 'channel first' convention). The previous formula remains valid:

$$\nabla_\theta \mathcal{L}(\theta) = v J_\theta \hat{s} = v \left([D_x, D_y] \odot \left(1_b \otimes 1_c \otimes [X, Y](J_\theta L_\theta^T)\right)\right). \tag{A.4}$$

With the following difference:

- $v = \nabla_{\hat{s}} \mathcal{L}(\hat{s}) \in \mathbb{R}^{b\times c\times n^2}$ is the gradient of the loss with respect to a batch of multiple channels images.
- $[D_x, D_y] \in \mathbb{R}^{b\times c\times n^2\times 2}$ are the stacked image derivatives of $\hat{s}$, computed in parallel on whole batch and all the channels. Notice that it can be done in a single Tensorflow operation using ' tf.nn.conv2d ' and a custom (nontrainable) kernel that contains the Sobel filter (Sobel et al., 1968) coefficients.
- $1_b \otimes 1_c \in \mathbb{R}^{b\times c}$ is a *broadcasting* operation, used to make $1_b \otimes 1_c \otimes [X, Y](J_\theta L_\theta^T)$ a tensor of shape $b \times c \times n^2 \times 2 \times d$ by duplicating bc times the tensor $[X, Y](J_\theta L_\theta^T)$.
- the batched vector-matrix product $\odot$ is now batched over $b \times c \times n^2$ dimensions.

A.1.2 Practical implementation

There is three ways to implement the computation of $\nabla_\theta \mathcal{L}(\theta)$.

- Let Autodiff handle the whole thing by defining $\hat{s}$ with differentiable operations. This is actually the case for $\hat{s} = [f(M_\theta^{-1}(x(i), y(j)))]_{0 \leq i,j \leq n-1}$. However the function f must be defined cleverly to keep the precision and the runtime reasonable.
- Compute $\nabla_\theta \mathcal{L}(\theta)$ "by-hand" using the above formula, which is easy to code using 'tf.einsum' and 'tf.nn.conv2d'. Higher order derivatives (with respect to model weights w) will be computed cheaply since v is the only vector that depends of neural network weights.
- Use a combination of both ideas by creating a 'tf.custom_gradient' decorator for the rotation operation.

Appendix B

Lipschitzness with respect to parameters

B.1 Proofs of the main result

The informal Theorem 1 requires some tools that we introduce below.

Additionnal hypothesis for GNP networks. We introduce convenient assumptions for the purpose of obtaining tight bounds in Algorithm 9.

Assumption 2 (Bounded biases)**.** *We assume there exists $B > 0$ such that that for all biases b_d we have $\|b_d\| \leq B$. Observe that the ball $\{\|b\|_2 \leq B\}$ of radius B is a* ***convex*** *set.*

Assumption 3 (Zero preserving activation)**.** *We assume that the activation fulfills $\sigma(\mathbf{0}) = \mathbf{0}$. When σ is S-Lipschitz this implies $\|\sigma(x)\| \leq S\|x\|$ for all x. Examples of activations fulfilling this constraints are ReLU, Groupsort, GeLU, ELU, tanh. However it does not work with sigmoid or softplus.*

We also propose the assumption 4 for convenience and exhaustivity.

Assumption 4 (Bounded activation)**.** *We assume it exists $G > 0$ such that for every $x \in \mathcal{X}$ and every $1 \leq d \leq D+1$ we have:*

$$\|h_d\| \leq G \quad \textit{and} \quad \|z_d\| \leq G. \tag{B.1}$$

Note that this assumption is implied by requirement 2, assumption 2-3, as illustrated in proposition 13.

In practice assumption 4 can be fulfilled with the use of input clipping and bias clipping, bounded activation functions, or layer normalization. This assumption can be used as a "shortcut" in the proof of the main theorem, to avoid the "propagation of input bounds" step.

B.1.1 Main result

We rephrase in a rigorous manner the informal theorem of section 6.3.1. In order to simplify the notations, we use $X := X_0$ in the following.

Proposition 13. Norm of intermediate activations. *Under requirement 2, assumptions 2-3 we have:*

$$\|h_t\| \leq S\|z_t\| \leq \begin{cases} (US)^t \left(X - \frac{SB}{1-SU}\right) + \frac{SB}{1-SU} & \text{if } US \neq 1, \\ SX + tSB & \text{otherwise.} \end{cases} \tag{B.2}$$

In particular if there are no biases, i.e if $B = 0$, then $\|h_t\| \leq S\|z_t\| \leq SX$.

Proposition 13 can be used to replace assumption 4.

Proposition 14. Lipschitz constant of dense Lipschitz networks with respect to parameters. *Let $f(\cdot,\cdot)$ be a Lipschitz neural network. Under requirement 2, assumptions 2-3 we have for every $1 \leq t \leq T+1$:*

$$\|\frac{\partial f(\theta,x)}{\partial b_t}\|_2 \leq (SU)^{T+1-t}, \tag{B.3}$$

$$\|\frac{\partial f(\theta,x)}{\partial W_t}\|_2 \leq (SU)^{T+1-t}\|h_{t-1}\|. \tag{B.4}$$

In particular, for every $x \in \mathcal{X}$, the function $\theta \mapsto f(\theta,x)$ is Lipschitz bounded.

Proposition 14 suggests that the scale of the activation $\|h_t\|$ must be kept under control for the gradient scales to distribute evenly along the computation graph. It can be easily extended to a general result on the *per sample* gradient of the loss, in theorem 3.

Theorem 3. Bounded loss gradient for dense Lipschitz networks. *Assume the predictions are given by a Lipschitz neural network f:*

$$\hat{y} := f(\theta,x). \tag{B.5}$$

Under requirements 1-2, assumptions 2-3, there exists a $K > 0$ for all $(x,y,\theta) \in \mathcal{X} \times \mathcal{Y} \times \Theta$ the loss gradient is bounded:

$$\|\nabla_\theta \mathcal{L}(\hat{y},y)\|_2 \leq K. \tag{B.6}$$

Let $\alpha = SU$ be the maximum spectral norm of the Jacobian between two consecutive layers.

If $\alpha = 1$ then we have:

$$K = \mathcal{O}\left(LX + L\sqrt{T} + LSX\sqrt{T} + L\sqrt{BX}ST + LBST^{3/2}\right). \tag{B.7}$$

The case $S = 1$ is of particular interest since it covers most activation function (i.e ReLU, GroupSort):

$$K = \mathcal{O}\left(L\sqrt{T} + LX\sqrt{T} + L\sqrt{BX}T + LBT^{3/2}\right). \tag{B.8}$$

Further simplification is possible if we assume $B = 0$, i.e a network without biases:

$$K = \mathcal{O}\left(L\sqrt{T}(1+X)\right). \tag{B.9}$$

If $\alpha > 1$ then we have:

$$\|\nabla_\theta \mathcal{L}(\hat{y},y)\|_2 = \mathcal{O}\left(L\frac{\alpha^T}{\alpha-1}\left(\sqrt{T}(\alpha X + SB) + \frac{\alpha(SB+\alpha)}{\sqrt{\alpha^2-1}}\right)\right). \tag{B.10}$$

Once again $B = 0$ (network with no bias) leads to useful simplifications:

$$\|\nabla_\theta \mathcal{L}(\hat{y},y)\|_2 = \mathcal{O}\left(L\frac{\alpha^{T+1}}{\alpha-1}\left(\sqrt{T}X + \frac{\alpha}{\sqrt{\alpha^2-1}}\right)\right). \tag{B.11}$$

*We notice that when $\alpha \gg 1$ there is an **exploding gradient** phenomenon where the upper bound become vacuous.*

If $\alpha < 1$ then we have:

$$\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2 = \mathcal{O}\left(L\alpha^T\left(X\sqrt{T} + \frac{1}{(1-\alpha^2)}\left(\sqrt{\frac{XSB}{\alpha^T}} + \frac{SB}{\sqrt{(1-\alpha)}}\right)\right) + \frac{L}{(1-\alpha)\sqrt{1-\alpha}}\right). \tag{B.12}$$

For network without biases we get:

$$\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2 = \mathcal{O}\left(L\alpha^T X\sqrt{T} + \frac{L}{\sqrt{(1-\alpha)^3}}\right). \tag{B.13}$$

*The case $\alpha \ll 1$ is a **vanishing gradient** phenomenon where $\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2$ is now independent of the depth T and of the input scale X.*

Proof. The control of gradient implicitly depend on the scale of the output of the network at every layer, hence it is crucial to control the norm of each activation.

Lemma 2 (Bounded activations)**.** *If $US \neq 1$ for every $1 \leq t \leq T+1$ we have:*

$$\|z_t\| \leq U^t S^{t-1}\left(X - \frac{SB}{1-SU}\right) + \frac{B}{1-SU}. \tag{B.14}$$

If $US = 1$ we have:

$$\|z_t\| \leq X + tB. \tag{B.15}$$

In every case we have $\|h_t\| \leq S\|z_t\|$.

Lemma proof. From assumption 3, if we assume that σ is S-Lipschitz, we have:

$$\|h_t\| = \|\sigma(z_t)\| = \|\sigma(z_t) - \sigma(\mathbf{0})\| \leq \mathbf{S}\|\mathbf{z_t}\|. \tag{B.16}$$

Now, observe that:

$$\|z_{t+1}\| = \|W_{t+1}h_t + b_{t+1}\| \leq \|W_{t+1}\|\|h_t\| + \|b_{t+1}\| \leq US\|z_t\| + B. \tag{B.17}$$

Let $u_1 = UX + B$ and $u_{t+1} = SUu_t + B$ be a linear recurrence relation. The translated sequence $u_t - \frac{B}{1-SU}$ is a geometric progression of ratio SU, hence $u_t = (SU)^{t-1}(UX + B - \frac{B}{1-SU}) + \frac{B}{1-SU}$. Finally we conclude that by construction $\|z_t\| \leq u_t$. ∎

The activation jacobians can be bounded by applying the chainrule. The recurrence relation obtained is the one automatically computed with back-propagation.

Lemma 3 (Bounded activation derivatives)**.** *For every $T+1 \geq s \geq t \geq 1$ we have:*

$$\|\frac{\partial z_s}{\partial z_t}\| \leq (SU)^{s-t}. \tag{B.18}$$

Lemma proof. The chain rule expands as:

$$\frac{\partial z_s}{\partial z_t} = \frac{\partial z_s}{\partial h_{s-1}}\frac{\partial h_{s-1}}{\partial z_{s-1}}\frac{\partial z_{s-1}}{\partial z_t}. \tag{B.19}$$

From Cauchy-Schwartz inequality we get:

$$\|\frac{\partial z_s}{\partial z_t}\| \leq \|\frac{\partial z_s}{\partial h_{s-1}}\| \cdot \|\frac{\partial h_{s-1}}{\partial z_{s-1}}\| \cdot \|\frac{\partial z_{s-1}}{\partial z_t}\|. \tag{B.20}$$

Since σ is S-Lipschitz, and $\|W_s\| \leq U$, and by observing that $\|\frac{\partial z_t}{\partial z_t}\| = 1$ we obtain by induction that:

$$\|\frac{\partial h_s}{\partial h_t}\| \leq (SU)^{s-t}. \tag{B.21}$$

■

The derivatives of the biases are a textbook application of the chainrule.

Lemma 4 (Bounded bias derivatives)**.** *For every t we have:*

$$\|\nabla_{b_t}\mathcal{L}(\hat{y}, y)\| \leq L(SU)^{T+1-t}. \tag{B.22}$$

Lemma proof. The chain rule yields:

$$\nabla_{b_t}\mathcal{L}(\hat{y}, y) = (\nabla_{\hat{y}}\mathcal{L}(\hat{y}, y))\frac{\partial z_{T+1}}{\partial z_t}\frac{\partial z_t}{\partial b_t}. \tag{B.23}$$

Hence we have:

$$\|\nabla_{b_t}\mathcal{L}(\hat{y}, y)\| = \|\nabla_{\hat{y}}\mathcal{L}(\hat{y}, y)\| \cdot \|\frac{\partial z_{T+1}}{\partial z_t}\| \cdot \|\frac{\partial z_t}{\partial b_t}\|. \tag{B.24}$$

We conclude with Lemma 3 that states $\|\frac{\partial z_{T+1}}{\partial z_t}\| \leq (US)^{T+1-t}$, with requirement 1 that states $\|\nabla_{\hat{y}}\mathcal{L}(\hat{y}, y)\| \leq L$ and by observaing that $\|\frac{\partial z_t}{\partial b_t}\| = 1$. ■

We can now bound the derivative of the affine weights:

Lemma 5 (Bounded weight derivatives)**.** *For every $T + 1 \geq t \geq 2$ we have:*

$$\|\nabla_{W_t}\mathcal{L}(\hat{y}, y)\| \leq L(SU)^T\left(X - \frac{SB}{1 - SU}\right) + L(SU)^{T+1-t}\frac{SB}{1 - SU} \text{ when } SU \neq 1, \tag{B.25}$$

$$\|\nabla_{W_t}\mathcal{L}(\hat{y}, y)\| \leq LS\left(X + (t-1)B\right) \text{ when } SU = 1. \tag{B.26}$$

$$\tag{B.27}$$

In every case:

$$\|\nabla_{W_1}\mathcal{L}(\hat{y}, y)\| \leq L(SU)^T X. \tag{B.28}$$

Lemma proof. We proceed like in the proof of Lemma 4 and we get:

$$\|\nabla_{W_t}\mathcal{L}(\hat{y}, y)\| \leq \|\nabla_{\hat{y}}\mathcal{L}(\hat{y}, y)\| \cdot \|\frac{\partial z_{T+1}}{\partial z_t}\| \cdot \|\frac{\partial z_t}{\partial W_t}\|. \tag{B.29}$$

Which then yields:

$$\|\nabla_{W_t}\mathcal{L}(\hat{y}, y)\| \leq L(SU)^{T+1-t} \cdot \|\frac{\partial z_t}{\partial W_t}\|. \tag{B.30}$$

Now, for $T + 1 \geq t \geq 1$, according to Lemma 2 we either have:

$$\|\frac{\partial z_t}{\partial W_t}\| \leq \|h_{t-1}\| \leq S\|z_{t-1}\| = (SU)^{t-1}\left(X - \frac{SB}{1 - SU}\right) + \frac{SB}{1 - SU}, \tag{B.31}$$

or, when $US = 1$:

$$\|\frac{\partial z_t}{\partial W_t}\| \leq \|h_{t-1}\| = S\|z_{t-1}\| = SX + (t-1)SB \text{ if } t \geq 2, \tag{B.32}$$

$$\|\frac{\partial z_t}{\partial W_t}\| \leq X \text{ otherwise.} \tag{B.33}$$

■

Now, the derivatives of the loss with respect to each type of parameter (i.e W_t or b_t) are know, and they can be combined to retrieve the overall gradient vector.

$$\theta = \{(W_1, b_1), (W_2, b_2), \dots (W_{T+1}, b_{T+1})\}. \tag{B.34}$$

We introduce $\alpha = SU$.

Case $\alpha = 1$. The resulting norm is given by the series:

$$\begin{aligned}
\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2^2 &= \sum_{t=1}^{T+1} \|\nabla_{b_t} \mathcal{L}(\hat{y}, y)\|_2^2 + \|\nabla_{W_t} \mathcal{L}(\hat{y}, y)\|_2^2 && \text{(B.35)}\\
&\leq L^2 \left((1 + X^2) + \sum_{t=2}^{T+1} (1 + (SX + (t-1)SB)^2) \right) && \text{(B.36)}\\
&\leq L^2 \left(1 + X^2 + \sum_{u=1}^{T} (1 + (SX + uSB)^2) \right) && \text{(B.37)}\\
&\leq L^2 \left(1 + X^2 + \sum_{u=1}^{T} (1 + S^2(X^2 + +2uBX + u^2B^2)) \right) && \text{(B.38)}\\
&\leq L^2 \left(1 + X^2 + T(1 + S^2X^2) + S^2BXT(T+1) + S^2B^2 \frac{T(T+1)(2T+1)}{6} \right). && \text{(B.39)}
\end{aligned}$$

Finally:

$$\begin{aligned}
\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2 &= \mathcal{O}(L\sqrt{X^2 + T + TS^2X^2 + BS^2XT^2 + B^2S^2T^3}) && \text{(B.40)}\\
&= \mathcal{O}\left(LX + L\sqrt{T} + LSX\sqrt{T} + L\sqrt{BX}ST + LBST^{3/2} \right). && \text{(B.41)}
\end{aligned}$$

This upper bound depends (asymptotically) linearly of $L, X, S, B, T^{3/2}$, when other factors are kept fixed to non zero value.

Case $\alpha \neq 1$. We introduce $\beta = \frac{SB}{1-\alpha}$.

$$\begin{aligned}
\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2^2 &= \sum_{t=1}^{T+1} \|\nabla_{b_t} \mathcal{L}(\hat{y}, y)\|_2^2 + \|\nabla_{W_t} \mathcal{L}(\hat{y}, y)\|_2^2 && \text{(B.42)}\\
&\leq L^2 \left(\alpha^{2T} \sum_{t=1}^{T+1} (((X - \beta) + \alpha^{1-t}\beta)^2 + \alpha^{2-2t}) \right) && \text{(B.43)}\\
&\leq L^2\alpha^{2T} \left(\sum_{u=0}^{T} (((X - \beta)^2 + 2(X - \beta)\alpha^{-u}\beta + \alpha^{-2u}\beta^2) + \alpha^{-2u}) \right) && \text{(B.44)}\\
&\leq L^2\alpha^{2T} \left((T+1)(X - \beta)^2 + 2(X - \beta)\beta \sum_{u=0}^{T} \alpha^{-u} + (\beta^2 + 1) \sum_{u=0}^{T} \alpha^{-2u} \right) && \text{(B.45)}\\
&\leq L^2\alpha^{2T} \left((T+1)(X - \beta)^2 + 2(X - \beta)\beta \frac{\alpha - (\frac{1}{\alpha})^T}{\alpha - 1} + (\beta^2 + 1) \frac{\alpha^2 - (\frac{1}{\alpha^2})^T}{\alpha^2 - 1} \right). && \text{(B.46)}
\end{aligned}$$

Finally:

$$\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2 \leq L\alpha^T \sqrt{(T+1)(X-\beta)^2 + 2(X-\beta)\beta\frac{\alpha - (\frac{1}{\alpha})^T}{\alpha - 1} + (\beta^2+1)\frac{\alpha^2 - (\frac{1}{\alpha^2})^T}{\alpha^2 - 1}}. \tag{B.47}$$

Now, the situation is a bit different for $\alpha < 1$ and $\alpha > 1$. One case corresponds to exploding gradient, and the other to vanishing gradient.

When $\alpha < 1$ we necessarily have $\beta > 0$, hence we obtain a crude upper-bound:

$$\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2 = \mathcal{O}\left(L\alpha^T\left(X\sqrt{T} + \frac{1}{(1-\alpha^2)}\left(\sqrt{\frac{XSB}{\alpha^T}} + \frac{SB}{\sqrt{(1-\alpha)}}\right)\right) + \frac{L}{(1-\alpha)\sqrt{1-\alpha}}\right). \tag{B.48}$$

Once again $B = 0$ (network with no bias) leads to useful simplifications:

$$\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2 = \mathcal{O}\left(L\alpha^T X\sqrt{T} + \frac{L}{\sqrt{(1-\alpha)^3}}\right). \tag{B.49}$$

This is a typical case of vanishing gradient since when $T \gg 1$ the upper bound does not depend on the input scale X anymore.

Similarly, we can perform the analysis for $\alpha > 1$, which implies $\beta < 0$, yielding another bound:

$$\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2 = \mathcal{O}\left(L\frac{\alpha^T}{\alpha - 1}\left(\sqrt{T}(\alpha X + SB) + \frac{\alpha(SB+\alpha)}{\sqrt{\alpha^2 - 1}}\right)\right). \tag{B.50}$$

Without biases we get:

$$\|\nabla_\theta \mathcal{L}(\hat{y}, y)\|_2 = \mathcal{O}\left(L\frac{\alpha^{T+1}}{\alpha - 1}\left(\sqrt{T}X + \frac{\alpha}{\sqrt{\alpha^2 - 1}}\right)\right). \tag{B.51}$$

We recognize an exploding gradient phenomenon due to the α^T term. □

Propositions 13 and 14 were introduced for clarity. They are a simple consequence of the Lemmas 2-4-5 used in the proof of Theorem 3.

The informal theorem of section 6.3.1 is based on the bounds of theorem 3, that have been simplified. Note that the definition of network differs slightly: in definition 6 the activations and the affines layers are considered independent and indexed differently, while the theoretical framework merge them into z_t and h_t respectively, sharing the same index t. This is without consequences once we realize that if $K = U = S$ and $2T = D$ then $(US)^2 = \alpha^2 = K^2$ leads to $\alpha^{2T} = K^D$. The leading constant factors based on α value have been replaced by 1 since they do not affect the asymptotic behavior.

B.2 Spectral bounds of layers

B.2.1 Lipschitz constants of common loss functions

This section contains the proofs related to the content of Table B.1. Our framework wraps over some losses found in deel-lip library, that are wrapped by our framework to provide Lipschitz constant automatically during backpropagation for bounds.

Loss	Hyper-parameters	$\mathcal{L}(\hat{y}, y)$	Lipschitz bound L
Softmax Cross-entropy	temperature $\tau > 0$	$y^T \log \text{softmax}(\hat{y}/\tau)$	$\sqrt{2}/\tau$
Cosine Similarity	bound $X_{\min} > 0$	$\frac{y^T \hat{y}}{\max(\|\hat{y}\|, X_{\min}}$	$1/X_{\min}$
Multiclass Hinge	margin $m > 0$	$\{\max(0, \frac{m}{2} - \hat{y}_i.y_i)\}_{1 \leq i \leq K}$	1
Kantorovich-Rubenstein	N/A	$\{\hat{y}, y\}$	1
Hinge Kantorovich-Rubenstein	margin $m > 0$ regularization $\alpha > 0$	$\alpha.\mathcal{L}_{MH}(\hat{y}, y) + \mathcal{L}_{MKR}(\hat{y}, y)$	$1 + \alpha$

Table B.1: Lipschitz constant of common supervised classification losses used for the training of Lipschitz neural networks with k classes. Proofs in Section B.2.1.

Multiclass Hinge This loss, with min margin m is computed in the following manner for a one-hot encoded ground truth vector y and a logit prediction $\hat{y}$:

$$\mathcal{L}_{MH}(\hat{y}, y) = \{\max(0, \frac{m}{2} - \hat{y}_1.y_1), ..., \max(0, \frac{m}{2} - \hat{y}_k.y_k)\}.$$

And $\|\frac{\partial}{\partial y}\mathcal{L}_{MH}(\hat{y}, y)\|_2 \leq \|\hat{y}\|_2$. Therefore $L_H = 1$.

Multiclass Kantorovich Rubenstein This loss, is computed in a one-versus all manner, for a one-hot encoded ground truth vector y and a logit prediction $\hat{y}$:

$$\mathcal{L}_{MKR}(\hat{y}, y) = \{\hat{y}_1 - y_1, \ldots, \hat{y}_k - y_k)\}.$$

Therefore, by differentiating, we also get $L_{KR} = 1$.

Multiclass Hinge - Kantorovitch Rubenstein This loss, is computed in the following manner for a one-hot encoded ground truth vector y and a logit prediction $\hat{y}$:

$$\mathcal{L}_{MHKR}(\hat{y}, y) = \alpha\mathcal{L}_{MH}(\hat{y}, y) + \mathcal{L}_{MKR}(\hat{y}, y).$$

By linearity we get $L_{HKR} = \alpha + 1$.

Cosine Similarity Cosine Similarity is defined in the following manner element-wise :

$$\mathcal{L}_{CS}(\hat{y}, y) = \frac{\hat{y}^T y}{\|\hat{y}\|_2 \|y\|_2}.$$

And y is one-hot encoded, therefore $\mathcal{L}_{CS}(\hat{y}, y) = \frac{\hat{y}_i}{\|\hat{y}\|_2}$. Therefore, the Lipschitz constant of this loss is dependant on the minimum value of $\hat{y}$. A reasonable assumption would be $\forall x \in \mathcal{D}$: $X_{min} \leq \|x\|_2 \leq X_{max}$. Furthermore, if the networks are Norm Preserving with factor K, we ensure that:

$$KX_{min} \leq \|\hat{y}\|_2 \leq KX_{max}.$$

Which yields: $L_{CS} = \frac{1}{KX_{min}}$. The issue is that the exact value of K is never known in advance since Lipschitz networks are rarely purely Norm Preserving in practice due to various effects (lack of tightness in convolutions, or rectangular matrices that can not be perfectly orthogonal).
Realistically, we propose the following loss function in replacement:

$$\mathcal{L}_{K-CS}(\hat{y}, y) = \frac{\hat{y}_i}{\max(KX_{min}, \|\hat{y}\|_2)}.$$

Layer	Hyper parameters	$\|\frac{\partial f_t(\theta_t,x)}{\partial \theta_t}\|_2$
1-Lipschitz dense	none	1
Convolution	window s	$\sqrt{s}$
RKO convolution	window s image size $H \times W$	$\sqrt{1/((1-\frac{(h-1)}{2H})(1-\frac{(w-1)}{2W}))}$

Table B.2: Lipschitz constant with respect to parameters in common Lipschitz layers. We report only the multiplicative factor that appears in front of the input norm $\|x\|_2$.

Where K is an input given by the user, therefore enforcing $L_{K-CS} = \frac{1}{KX_{min}}$.

Categorical Cross-entropy from logits The logits are mapped into the probability simplex with the *Softmax* function $\mathbb{R}^K \to (0,1)^K$. We also introduce a temperature parameter $\tau > 0$, which hold signifance importance in the accuracy/robustness tradeoff for Lipschitz networks as observed by Béthune et al. (2022). We assume the labels are discrete, or one-hot encoded: we do not cover the case of label smoothing.

$$S_j = \frac{\exp(\tau \hat{y}_j)}{\sum_i \exp(\tau \hat{y}_i)}. \tag{B.52}$$

We denote the prediction associated to the true label j^+ as S_{j^+}. The loss is written as:

$$\mathcal{L}(\hat{y}) = -\log(S_{j^+}). \tag{B.53}$$

Its gradient with respect to the logits is:

$$\nabla_{\hat{y}} \mathcal{L} = \begin{cases} \tau(S_{j^+} - 1) & \text{if } j = j^+, \\ \tau S_j & \text{otherwise} \end{cases} \tag{B.54}$$

The temperature factor τ is a multiplication factor than can be included in the loss itself, by using $\frac{1}{\tau}\mathcal{L}$ instead of $\mathcal{L}$. This formulation has the advantage of facilitating the tuning of the learning rate: this is the default implementation found in deel-lip library. The gradient can be written in vectorized form:

$$\nabla_{\hat{y}} \mathcal{L} = S - 1_{\{j=j^+\}}.$$

By definition of Softmax we have $\sum_{j \neq j^+} S_j^2 \leq 1$. Now, observe that $S_j \in (0,1)$, and as a consequence $(S_{j^+} - 1)^2 \leq 1$. Therefore $\|\nabla_{\hat{y}} \mathcal{L}\|_2^2 = \sum_{j \neq j^+} S_j^2 + (S_{j^+} - 1)^2 \leq 2$. Finally $\|\nabla_{\hat{y}} \mathcal{L}\|_2 = \sqrt{2}$ and $L_{CCE} = \sqrt{2}$.

B.2.2 Layer bounds

The Lipschitz constant (with respect to input) of each layer of interest is summarized in table B.3, while the Lipschitz constant with respect to parameters is given in table B.2.

Dense layers

Below, we illustrate the basic properties of Lipschitz constraints and their consequences for gradient bounds computations. While for dense layers the proof is straightforward, the main ideas can be re-used for all linear operations which includes the convolutions and the layer centering.

Layer	Hyper parameters	$\|\frac{\partial f_t(\theta_t,x)}{\partial x}\|_2$
Add bias	none	1
1-Lipschitz dense	none	1
RKO convolution	none	1
Layer centering	none	1
Residual block	none	2
ReLU, GroupSort softplus, sigmoid, tanh	none	1

Table B.3: Lipschitz constant with respect to intermediate activations.

Property 9. Gradients for dense Lipschitz networks. *Let $x \in \mathbb{R}^C$ be a data-point in space of dimensions $C \in \mathbb{N}$. Let $W \in \mathbb{R}^{C \times F}$ be the weights of a dense layer with F features outputs. We bound the spectral norm of the Jacobian as*

$$\left\| \frac{\partial (W^T x)}{\partial W} \right\|_2 \leq \|x\|_2. \tag{B.55}$$

Proof. Since $W \mapsto W^T x$ is a linear operator, its Lipschitz constant is exactly the spectral radius:

$$\frac{\|W^T x - W'^T x\|_2}{\|W - W'\|_2} = \frac{\|(W - W')^T x\|_2}{\|W - W'\|_2} \leq \frac{\|W - W'\|_2 \|x\|_2}{\|W - W'\|_2} = \|x\|_2.$$

Finally, observe that the linear operation $x \mapsto W^T x$ is differentiable, hence the spectral norm of its Jacobian is equal to its Lipschitz constant with respect to l_2 norm. □

Convolutions

Property 10. Gradients for convolutional Lipschitz networks. *Let $x \in \mathbb{R}^{S \times C}$ be an data-point with channels $C \in \mathbb{N}$ and spatial dimensions $S \in \mathbb{N}$. In the case of a time serie S is the length of the sequence, for an image $S = HW$ is the number of pixels, and for a video $S = HWN$ is the number of pixels times the number of frames. Let $\Psi \in \mathbb{R}^{s \times C \times F}$ be the weights of a convolution with:*

- *window size $s \in \mathbb{N}$ (e.g $s = hw$ in 2D or $s = hwn$ in 3D),*
- *with C input channels,*
- *with $F \in \mathbb{N}$ output channels.*
- *we don't assume anything about the value of strides. Our bound is typically tighter for strides=1, and looser for larger strides.*

We denote the convolution operation as $(\Psi * \cdot) : \mathbb{R}^{S \times C} \to \mathbb{R}^{S \times F}$ with either zero padding, either circular padding, such that the spatial dimensions are preserved. Then the Jacobian of convolution operation with respect to parameters is bounded:

$$\|\frac{\partial (\Psi * x)}{\partial \Psi}\|_2 \leq \sqrt{s}\|x\|_2. \tag{B.56}$$

Proof. Let $y = \Psi * x \in \mathbb{R}^{S\times F}$ be the output of the convolution operator. Note that y can be uniquely decomposed as sum of output feature maps $y = \sum_{f=1}^{F} y^f$ where $y^f \in \mathbb{R}^{S\times F}$ is defined as:

$$\begin{cases} (y^f)_{if} = y_{if} & \text{for all } 1 \le i \le S, \\ (y^f)_{ij} = 0 & \text{if } j \neq f. \end{cases}$$

Observe that $(y^f)^T y^{f'} = 0$ whenever $f \neq f'$. As a consequence Pythagorean theorem yields $\|y\|_2^2 = \sum_{f=1}^{F} \|y^f\|_2^2$. Similarly we can decompose each output feature map as a sum of pixels $y^f = \sum_{p=1}^{S} y^{pf}$. where $y^{pf} \in \mathbb{R}^{S\times F}$ fulfill:

$$\begin{cases} (y^{pf})_{ij} = 0 & \text{if } i \neq p, j \neq f, \\ (y^{pf})_{pf} = y_{pf} & \text{otherwise.} \end{cases}$$

Once again Pythagorean theorem yields $\|y^f\|_2^2 = \sum_{p=1}^{S} \|y^{pf}\|_2^2$. It remains to bound y^{pf} appropriately. Observe that by definition:

$$y^{pf} = (\Psi * x)_{pf} = (\Psi^f)^T x^p[s].$$

where $\Psi^f \in \mathbb{R}^{s\times C}$ is a slice of Ψ corresponding to output feature map f, and $x^p[s] \in \mathbb{R}^{s\times C}$ denotes the patch of size s centered around input element p. For example, in the case of images with $s = 3\times 3$, p are the coordinates of a pixel, and $x^p[s]$ are the input feature maps of 3×3 pixels around it. We apply Cauchy-Schwartz:

$$\|y^{pf}\|_2^2 \le \|\Psi^f\|_2^2 \times \|x^p[s]\|_2^2.$$

By summing over pixels we obtain:

$$\|y^f\|_2^2 \le \|\Psi^f\|_2^2 \sum_{p=1}^{S} \|x^p[s]\|_2^2, \tag{B.57}$$

$$\implies \|y\|_2^2 \le (\sum_{f=1}^{F} \|\Psi^f\|_2^2)(\sum_{p=1}^{S} \|x^p[s]\|_2^2), \tag{B.58}$$

$$\implies \|y\|_2^2 \le \|\Psi\|_2^2 \times \left(\sum_{p=1}^{S} \|x^p[s]\|_2^2\right). \tag{B.59}$$

The quantity of interest is $\sum_{p=1}^{S} \|x^p[s]\|_2^2$ whose squared norm is the squared norm of all the patches used in the computation. With zero or circular padding, the norm of the patches cannot exceed those of input image. Note that each pixel belongs to atmost s patches, and even exactly s patches when circular padding is used:

$$\sum_{p=1}^{S} \|x^p[s]\|_2^2 \le s \sum_{p=1}^{S} \|x_p\|_2^2 = s\|x\|_2^2.$$

Note that when strides>1 the leading multiplicative constant is typically smaller than s, so this analysis can be improved in future work to take into account strided convolutions. Since Ψ is a linear operator, its Lipschitz constant is exactly its spectral radius:

$$\frac{\|(\Psi * x) - (\Psi' * x)\|_2}{\|\Psi - \Psi'\|_2} = \frac{\|(\Psi - \Psi') * x\|_2}{\|\Psi - \Psi'\|_2} \le \frac{\sqrt{s}\|\Psi - \Psi'\|_2\|x\|_2}{\|\Psi - \Psi'\|_2} = \sqrt{s}\|x\|_2.$$

Finally, observe that the convolution operation $\Psi * x$ is differentiable, hence the spectral norm of its Jacobian is equal to its Lipschitz constant with respect to l_2 norm:

$$\|\frac{\partial(\Psi * x)}{\partial \Psi}\|_2 \leq \sqrt{s}\|x\|_2.$$

□

An important case of interest are the convolutions based on Reshaped Kernel Orthogonalization (RKO) method introduced by Li et al. (2019a). The kernel Ψ is reshaped into 2D matrix of dimensions $(sC \times F)$ and this matrix is orthogonalised. This is not sufficient to ensure that the operation $x \mapsto \Psi * x$ is orthogonal - however it is 1-Lipschitz and only *approximately* orthogonal under suitable re-scaling by $\mathcal{N} > 0$.

Corollary 5 (Loss gradient for RKO convolutions.)**.** *For RKO methods in 2D used in Serrurier et al. (2021), the convolution kernel is given by $\Phi = \mathcal{N}\Psi$ where Ψ is an orthogonal matrix (under RKO) and $\mathcal{N} > 0$ a factor ensuring that $x \mapsto \Phi * x$ is a 1-Lipschitz operation. Then, for RKO convolutions without strides we have:*

$$\|\frac{\partial(\Psi * x)}{\partial \Psi}\|_2 \leq \sqrt{\frac{1}{(1-\frac{(h-1)}{2H})(1-\frac{(w-1)}{2W})}}\|x\|_2. \tag{B.60}$$

where (H, W) are image dimensions and (h, w) the window dimensions. For large images with small receptive field (as it is often the case), the Taylor expansion in $h \ll H$ and $w \ll W$ yields a factor of magnitude $1 + \frac{(h-1)}{4H} + \frac{(w-1)}{4W} + \mathcal{O}(\frac{(w-1)(h-1)}{8HW}) \approx 1$.

Layer normalizations

Property 3. Bounded loss gradient for layer centering. *Layer centering is defined as $f(x) = x - (\frac{1}{n}\sum_{i=1}^{n} x_i)\mathbf{1}$ where $\mathbf{1}$ is a vector full of ones and acts as a "centering" operation along some channels (or all channels). Then the singular values of this linear operation are:*

$$\sigma_1 = 0, \quad \textit{and} \quad \sigma_2 = \sigma_3 = \ldots = \sigma_n = 1. \tag{2.7}$$

In particular $\|\frac{\partial f}{\partial x}\|_2 \leq 1$.

Proof. It is clear that layer normalization is an affine layer. Hence the spectral norm of its Jacobian coincides with its Lipschitz constant with respect to the input, which itself coincides with the spectral norm of f. The matrice M associated to f is symmetric and diagonally dominant since $|\frac{n-1}{n}| \geq \sum_{i=1}^{n-1} |\frac{-1}{n}|$. It follows that M is semi-definite positive. In particular all its eigenvalues $\lambda_1 \leq \ldots \leq \lambda_n$ are non negative. Furthermore they coincide with its singular values: $\sigma_i = \lambda_i$. Observe that for all $r \in \mathbb{R}$ we have $f(r\mathbf{1}) = \mathbf{0}$, i.e the operation is null on constant vectors. Hence $\lambda_1 = 0$. Consider the matrix $M - I$: its kernel is the eigenspace associated to eigenvalue 1. But the matrix $M - I = \frac{-1}{n}\mathbf{1}\mathbf{1}^T$ is a rank-one matrix. Hence its kernel is of dimension $n - 1$, from which it follows that $\lambda_2 = \ldots = \lambda_n = \sigma_2 \ldots = \sigma_n = 1$. □

MLP Mixer architecture

The MLP-mixer architecture introduced in Tolstikhin et al. (2021) consists of operations named *Token mixing* and *Channel mixing* respectively. For token mixing, the input feature is split in disjoint patches on which the same linear opration is applied. It corresponds to a convolution with a stride

Dataset	Samples	Features	δ	Validation AUROC ↑		Wallclock Runtime (s) ↓	
				DP-SGD	Clipless DP-SGD	DP-SGD	Clipless DP-SGD
ALOI	39,627	27	10^{-5}	**56.5**	56.2	159.1	**11.3**
campaign	32,950	62	10^{-5}	**90.0**	82.2	155.6	**11.8**
celeba	162,079	39	10^{-6}	**96.6**	96.5	41.1	**34.6**
census	239,428	500	10^{-6}	**93.3**	92.5	820.0	**79.8**
donors	495,460	10	10^{-6}	100.0	**100.0**	257.9	**90.2**
magic	15,216	10	10^{-5}	**90.7**	89.7	89.9	**56.0**
shuttle	39,277	9	10^{-5}	98.3	**99.4**	11.1	**6.8**
skin	196,045	3	10^{-6}	**100.0**	99.8	54.2	**40.2**
yeast	1,187	8	10^{-4}	66.8	**75.1**	22.1	**5.6**

Table B.4: Best validation AUROC values (in %) for models trained under (ϵ, δ)-DP privacy with $\epsilon = 1$, with DP-SGD and Clipless DP-SGD, on binary classification tasks of tabular data from Adbench datasets Han et al. (2022). We use a random 80/20% stratified split into train/val.

equal to the kernel size. convolutions on a reshaped input, where patches of pixels are "collapsed" in channel dimensions. Since the same linear transformation is applied on each patch, this can be interpreted as a block diagonal matrix whose diagonal consists of W repeated multiple times. More formally the output of Token mixing takes the form of $f(x) := [W^T x_1, W^T x_2, \dots W^T x_n]$ where $x = [x_1, x_2, \dots, x_n]$ is the input, and the x_i's are the patches (composed of multiple pixels). Note that $\|f(x)\|_2^2 \leq \sum_{i=1}^n \|W\|_2^2 \|x_i\|_2^2 = \|W\|_2^2 \sum_{i=1}^n \|x_i\|_2^2 = \|W\|_2^2 \|x\|_2^2$. If $\|W\|_2 = 1$ then the layer is 1-Lipschitz - it is even norm preserving. Same reasoning apply for Channel mixing. Therefore the MLP_Mixer architecture is 1-Lipchitz and the weight sensitivity is proportional to $\|x\|$.

Lipschitz MLP mixer: We adapted the original architecture in order to have an efficient 1-Lipschitz version with the following changes:

- Relu activations were replaced with GroupSort, allowing a better gradient norm preservation,
- Dense layers were replaced with their GNP equivalent,
- Skip connections are available (adding a 0.5 factor to the output in order to ensure 1-lipschitz condition) but architecture perform as well without these.

Finally the architecture parameters were selected as following:

1. The number of layer is reduced to a small value (between 1 and 4) to take advantage of the theoretical sensitivity bound.
2. The patch size and hidden dimension are selected to achieve a sufficiently expressive network (a patch size between 2 and 4 achieves sufficient accuracy without over-fitting, and a hidden dimension of 128-512 unlocks allows batch size).
3. The channel dim and token dimensions are chosen such that weight matrices are square matrices (exact gradient norm preservation property requires square matrices).

B.3 Experimental settings

B.3.1 Tabular data

The results are given in Table B.4.

Hyperparameter Tuning	Influence on utility	Influence on privacy leakage per step
Increasing Batch Size	Beneficial: decreases the sensitivity.	Detrimental: reduces the privacy amplification by subsampling.
Loss Gradient Clipping	Beneficial: tighter sensitivity bounds. Detrimental: biases the direction of the gradient.	No influence
Clipping Input Norms	Detrimental: destroy information, but may increases generalization	No influence

Figure B.1: **Hyperparameter table:** Here, we give insights on the influence of some hyperparameters on utility and privacy.

B.3.2 Pareto fronts

We rely on Bayesian optimization Snoek et al. (2012) with Hyper-band Li et al. (2017) heuristic for early stopping. The influence of some hyperparameters has to be highlighted to facilitate training with our framework, therefore we provide a table that provides insights into the effects of principal hyperparameters in Figure B.1. Most hyper-parameters extend over different scales (such as the learning rate), so they are sampled according to log-uniform distribution, to ensure fair covering of the search space. Additionnaly, the importance of the softmax cross-entropy temperature τ has been demonstrated in previous work Béthune et al. (2022).

The sweeps have been done on various architectures such as Lipschitz VGGs, Lipschitz ResNets and Lipschitz MLP_Mixer. We can also break down the results per architecture, in figure B.2. The MLP_Mixer architecture seems to yield the best results. This architecture is exactly GNP since the orthogonal linear transformations are applied on disjoint patches. To the contrary, VGG and Resnets are based on RKO convolutions which are not exactly GNP. Hence those preliminary results are compatible with our hypothesis that GNP layers should improve performance. Note that these results are expected to change as the architectures are further improved. It is also dependant of the range chosen for hyper-parameters. We do not advocate for the use of an architecture over another, and we believe many other innovations found in literature should be included before settling the question definitively.

For the vanilla implementation of DP-SGD we rely on Opacus library. We use the default configuration from the official tutorial on Cifar-10.

B.3.3 Configuration of the "speed" experiment

We detail below the environment version of each experiment, together with Cuda and Cudnn versions. We rely on a machine with 32GB RAM and a NVIDIA Quadro GTX 8000 graphic card with 48GB memory. The GPU uses driver version 495.29.05, cuda 11.5 (October 2021) and cudnn 8.2 (June 7, 2021). We use Python 3.8 environment.

- For Jax, we used jax 0.3.17 (Aug 31, 2022) with jaxlib 0.3.15 (July 23, 2022), flax 0.6.0 (Aug 17, 2022) and optax 1.4.0 (Nov 21, 2022).
- For Tensorflow, we used tensorflow 2.12 (March 22, 2023) with tensorflow_privacy 0.7.3 (September 1, 2021).
- For Pytorch, we used Opacus 1.4.0 (March 24, 2023) with Pytorch (March 15, 2023).

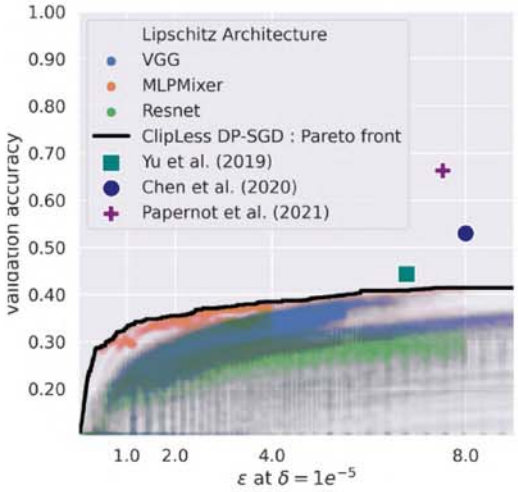


Figure B.2: **Accuracy/Privacy tradeoff on Cifar-10, split down per architecture used.** While some architectures seems to perform better than others, we don't advocate for the use of one over another. The results may not translate to all datasets, and may be highly dependant on the range chosen for hyper-parameters. While this figure provides valuable insights, identifying the best architecture is left for future works.

- For lip-dp we used deel-lip 1.4.0 (January 10, 2023) on Tensorflow 2.8 (May 23, 2022).

For this benchmark, we used among the most recent packages on pypi. However the latest version of tensorflow privacy could not be forced with pip due to broken dependencies. This issue arise in clean environments such as the one available in google colaboratory.

B.3.4 Drop-in replacement with Lipschitz networks in vanilla DP-SGD

To highlight the importance of the tight sensitivity bounds Δ_d obtained by our framework, we perform an ablation study by optimizing GNP networks using "vanilla" DP-SGD (with clipping), in Figure B.3a and B.3b.

Thanks to the gradient clipping of DP-SGD (see Algorithm 12), Lipschitz networks can be readily integrated in traditional DP-SGD algorithm with gradient clipping. The PGD algorithm is not mandatory: the back-propagation can be performed within the computation graph through iterations of Björck algorithm (used in RKO convolutions). This does not benefit from any particular speed-up over conventional networks - quite to the contrary there is an additional cost incurred by enforcing Lipschitz constraints in the graph. Some layers of deel-lip library have been recoded in Jax/Flax, and the experiment was run in Jax, since Tensorflow was too slow.

We use use the Total Amount of Noise (TAN) heuristic introduced in Sander et al. (2022) to heuristically tune hyper-parameters jointly. This ensures fair covering of the Pareto front.

B.3.5 Extended limitations

The main weakness of our approach is that it crucially rely on accurate computation of the sensitivity Δ. This task faces many challenges in the context of differential privacy: floating point arithmetic is not associative, and summation order can a have dramatic consequences regarding numerical stability Goldberg (1991). This is further amplified on the GPUs, where some operations

Algorithm 12 Differentially Private Stochastic Gradient Descent : **DP-SGD**

Input: Neural network architecture $f(\cdot, \cdot)$

Input: Initial weights θ_0, learning rate scheduling η_t, number of steps N, noise multiplier σ, L2 clipping value C .

1: **repeat**

2: **for all** $1 \leq t \leq N-1$ **do**

3: Sample a batch

$$\mathcal{B}_t = (x_1, y_1), (x_2, y_2), \ldots, (x_b, y_b).$$

4: Create microbatches, compute and clip the per-sample gradient of cost function:

$$\tilde{g}_{t,i} := \min(C, \|\nabla_{\theta_t} \mathcal{L}(\hat{y}_i, y_i)\|) \nabla_{\theta_t} \frac{\mathcal{L}(\hat{y}_i, y_i)}{\|\nabla_{\theta_t} \mathcal{L}(\hat{y}_i, y_i)\|}.$$

5: Perturb each microbatch with carefully chosen noise distribution $b \sim \mathcal{N}(0, \sigma C)$:

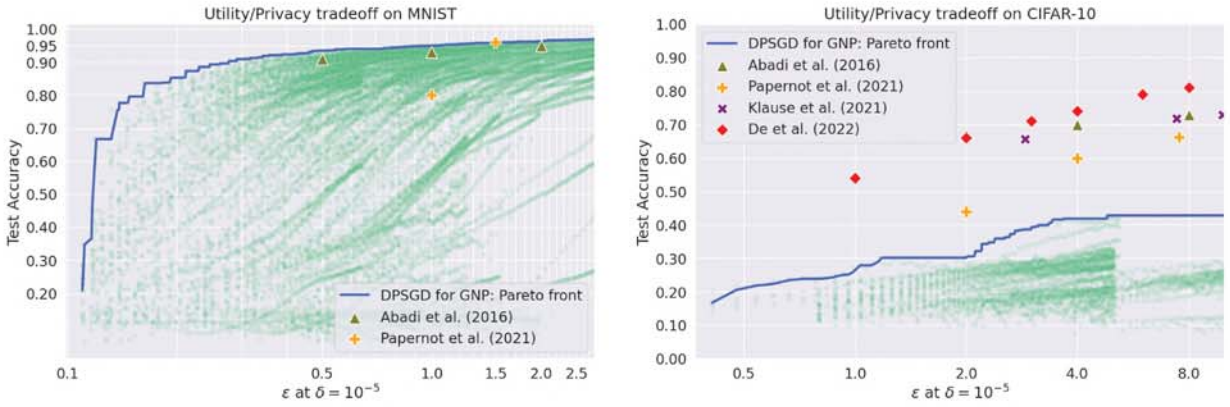
$$\hat{g}_{t,i} \leftarrow \tilde{g}_{t,i} + b_i.$$

6: Perform projected gradient step:

$$\theta_{t+1} \leftarrow \Pi(\theta_t - \eta_t \hat{g}_{t,i}).$$

7: **end for**

8: **until** privacy budget (ϵ, δ) has been reached.



(a) (b)

Figure B.3: **Privacy/utility trade-off for Gradient Norm Preserving networks trained under "vanilla" DP-SGD (with** gradient clipping). Each green dot corresponds to a single epoch of one of the runs. Trajectories that end abruptly are due to the automatic early stopping of unpromising runs. Note that clipping + orthogonalization have a high runtime cost, which severely limits the number of epochs reported.

are intrinsically non deterministic Jooybar et al. (2013). This well known issue is already present in vanilla DP-SGD algorithm. Our framework adds an additional point of failure: the upper bound of spectral Jacobian must be computed accurately. Hence Power Iteration must be run with sufficiently high number of iterations to ensure that the projection operator Π works properly. The (ϵ, δ)-DP certificates only hold under the hypothesis that all computations are correct, as numerical errors can induce privacy leakages. Hence we check empirically the effective norm of the gradient in the training loop at the end of each epoch. No certificate violations were reported during ours experiments, which suggests that the numerical errors can be kept under control.

Appendix C

Differentiable Gaussian Processes on Distributions

We let $\mathcal{P}(\Omega)$ be the set of probability measures over some compact space $\Omega \subset \mathbb{R}^d$. The goal is to address the "distribution regression problem" where the inputs are probability distributions and the output a real-valued observation:

$$Y_i = f_i^*(\mu_i) + \epsilon, \tag{C.1}$$

for $i = 1, \dots, n$ where $\mu_i \in \mathcal{P}(\Omega)$. The pairs (μ_i, Y_i) are i.i.d, $f_i^*(\mu_i)$ modelizes the conditional expectation of Y_i given μ_i, or equivalently $\mathbb{E}[\epsilon_i|\mu_i] = 0$. The goal is to learn the function f^*.

Warning C.1. Two-stage sampling.

There are two sources of randomness in the task. The first source, as often in learning, comes from the *first stage sampling* $(\mu_1, \mu_2, \dots, \mu_n) \sim \mathcal{P}(\mathbb{R}^d)^{\otimes n}$ itself. However, in many practical applications, the distributions μ_i are never observed directly. Rather, only empirical distribution $P_i \sim \mu_i^{\otimes N}$ are observed, with $P_i = \{X_{i,1}, X_{i,2}, \dots X_{i,N}\}$. This motivates the terminology *two-stage sampling*, with the first stage consisting of $(\mu_1, \mu_2, \dots, \mu_n)$ and the second one the sampling of $\{X_{i,1}, X_{i,2}, \dots X_{i,N}\}$ for each μ_i.

For more details on the theoretical framework, the reader should refer to Bachoc et al. (2023a) and Bachoc et al. (2023b).

Contents

C.1 Distribution Regression with Optimal Transport

In this section, we detail how to solve the *distribution regression* problem with a new Gaussian process whose kernel relies on regularized optimal transport.

C.1.1 Regularized optimal transport

For two distributions $P, Q \in \mathcal{P}(\Omega)$ we note $\Pi(P, Q)$ the set of probability measures over $\mathbb{R}^d \times \mathbb{R}^d$ with marginals P and Q respectively. We consider optimal transport problem with squared Euclidean cost $c(x, y) = \|x - y\|_2^2$. We consider optimal transport problem with entropic regularization, as in Cuturi (2013):

$$S_\epsilon(P, Q) = \min_{\pi \in \Pi(P,Q)} \int_{\Omega \times \Omega} \frac{1}{2}\|x - y\|^2 d\pi(x, y) + \epsilon H(\pi | P \otimes Q) \tag{C.2}$$

with

$$H(\alpha|\beta) = \int_\Omega \frac{d\alpha}{d\beta}(x) d\alpha(x) \tag{C.3}$$

the relative entropy, and $P \otimes Q$ the product measure.

The entropic regularization term modifies the linear term in OT (here a quadratic cost) into a strictly convex function. The minimization of Equation C.2 is achieved with Sinkhorn algorithm (see Peyré et al. (2017) and references therein). These properties ensure that a valid positive definite kernel can be built upon the solution of this OT problem. More precisely, we will rely on the dual formulation given by Genevay et al. (2018):

$$S_\epsilon(P, Q) = \sup_{f \in L_1(P), g \in \in L_1(Q)} Pf + Qg - \epsilon \mathbb{E}_{x \sim P, y \sim Q}[\exp \frac{1}{\epsilon}(f(x) + g(y) - \frac{1}{2}\|x - y\|_2^2)] + \epsilon. \tag{C.4}$$

This problem is itself a convex relaxation of the original's dual OT problem. The optimal potentials f and g are unique up to a (common) constant. They verify the *optimality conditions*:

$$\forall x \in \Omega, \quad \mathbb{E}_{y \sim Q}[\exp \frac{1}{\epsilon}(f(x) + g(y) - \frac{1}{2}\|x - y\|_2^2)] = 1, \tag{C.5}$$

$$\forall y \in \Omega, \quad \mathbb{E}_{x \sim P}[\exp \frac{1}{\epsilon}(f(x) + g(y) - \frac{1}{2}\|x - y\|_2^2)] = 1. \tag{C.6}$$

C.1.2 Building a PSD kernel with a reference measure

We consider a reference measure $\mathcal{U}$ on Ω. The idea is to solve the regularized OT problem between each P and $\mathcal{U}$, and to use the dual potential $g_{\mathbf{u}}^P$ as an *embedding* of the distribution P. Regardless of the support of P (discrete, continuous) the dual potential $g_{\mathbf{u}}^P$ only depends on the support of

$\mathcal{U}$. For arbitrary distributions $\mathcal{U}$, $g^P_{\mathbf{u}}$ is q function. Typically, the Hilbertian embeddings $P \mapsto g^P_{\mathbf{u}}$ considered theoretically are valued in infinite-dimensional Hilbert spaces. On the other hand, the numerical implementations of these embeddings map distributions to vectors. We shall refer to the dimensions of these vectors as the embeddings' dimensions. In our case, we will choose a discrete measure $\mathcal{U} = \sum_{i=1}^{|\mathbf{u}|} w_i \delta_{x_i}$ with $\mathbf{1^T w} = \mathbf{1}$, $\mathbf{w} \geq \mathbf{0}$ and δ_x the Dirac measure in x. In this case, the potential is a finite-dimensional vector $g^P_{\mathbf{u}} \in \mathbb{R}^q$.

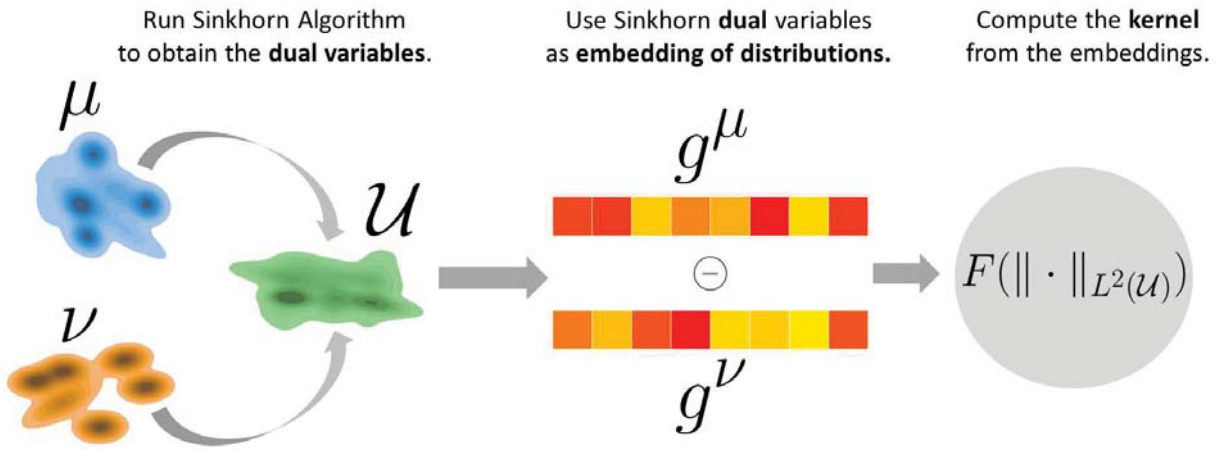
Finally, a kernel is built from these Hilbertian embeddings:

$$K_{\mathcal{U}}(P, Q) := F(\|g^P_{\mathbf{u}} - g^Q_{\mathbf{u}}\|), \tag{C.7}$$

where F is q function with some monotonicity constraints, which includes the well-known square exponential, power exponential and Matérn covariance functions (see Bachoc et al. (2020) and references therein). In the experiments we chose

$$K_{\mathcal{U}}(P, Q) := \sigma^2 \exp - \frac{\|g^P_{\mathbf{u}} - g^Q_{\mathbf{u}}\|}{2l}, \tag{C.8}$$

with length scale $l > 0$ and variance $\sigma > 0$. The method to build the kernel is illustrated below.



Remark C.1. Properties of the kernel.

The function $K_{\mathcal{U}}(P, Q)$ inherits several good properties from the regularized OT problem.

1. $K_{\mathcal{U}}(P, Q)$ is a valid **Positive Definite** kernel. It corresponds to an inner product in Hilbertian space. It can be used for kernel methods, including Support Vector Machines () or Gaussian Processes ().
2. It is **universal**. Said otherwise, the linear combinations

$$P \mapsto \sum_{i=1}^{n} \alpha_i K_{\mathcal{U}}(P, P_i) \tag{C.9}$$

   are dense in the set of real continuous functions $\mathcal{P}(\Omega) \to \mathbb{R}$ over distributions (for some topology). Therefore, this kernel is expressive enough for practical purposes.
3. It is **statistically consistent**, a key property when dealing with two-stage sampling. That means that the empirical kernel $K_{\mathcal{U}}(P_n, Q_n)$ converges to the population kernel

$K_{\mathcal{U}}(P, Q)$ as $n \to +\infty$.

C.1.3 Gaussian Processes on distribution

We recall below the framework of Gaussian Process (GP) modeling, for more details, the reader should refer to Rasmussen and Williams (2006). A GP indexed by E is a stochastic process entirely defined by its mean $m(x)$ and covariance function $k(x, x')$. In a GP, each of the observations follows a normal distribution conditioned to the other observations and parameters.

$$m(x) = \mathbb{E}[f(x)], \tag{C.10}$$
$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x')], \tag{C.11}$$

and the GP is written as

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')). \tag{C.12}$$

The GP defines a *posterior* probability over functions given observations X:

$$p(f|x) \propto p(x|f)p(f), \tag{C.13}$$

where we recognize the *prior* $p(f)$ and *likelihood* $p(x|f)$. The *marginal likelihood* of the observation y given data X is given by

$$p(y|X) = \int p(y|f, X)p(f|X)df \tag{C.14}$$

where the term "marginal" refers to the marginalization over functions f. For GP we have $f|X \sim \mathcal{N}(0, k)$ so we get

$$\log p(f|X) = -\frac{1}{2} f^T K^{-1} f - \frac{1}{2} \log |K| - \frac{2}{n} \log 2\pi. \tag{C.15}$$

Finally, the likelihood verifies $y|f \sim (0, K + \sigma^2 I)$, so we get

$$\log p(y|X) = -\frac{1}{2} y^T (K + \sigma^2 I)^{-1} y - \frac{1}{2} \log |K + \sigma^2 I| - \frac{2}{n} \log 2\pi. \tag{C.16}$$

Here we consider a covariance function defined by the kernel $K_{\mathcal{U}}(P, Q)$. We hyper-parameters θ of the kernel are optimized by maximizing the log marginal likelihood $\log p(y|X)$, with a gradient-based method. The derivative admits a closed form

$$\frac{\partial \log p_\theta(y|X)}{\partial \theta} = -\frac{1}{2} y^T K_\theta^{-1} \frac{\partial K_\theta}{\partial \theta} K_\theta^{-1} y - \frac{1}{2} \mathrm{Tr}(K_\theta^{-1} \frac{\partial K_\theta}{\partial \theta}). \tag{C.17}$$

This can be implemented in *Jax* with automatic support for Autodiff. In our case, the parameters θ are the triplet $(\mathbf{u}, \sigma, l)$. The inverted matrix K_θ^{-1} does not need to be computed. Instead, the operators $z_1 \mapsto K_\theta z_1$ can be used in indirect methods to solve the linear systems $K_\theta z_1 = y$ and $K_\theta z_2 = \frac{\partial K_\theta}{\partial \theta} z_1$, with the placeholders variables z_1, z_2 verify $z_1 := K_\theta^{-1} y$ and $z_2 = K_\theta^{-1} \frac{\partial K_\theta}{\partial \theta} z_1$.

C.2 Autodiff for fine-tuning of the reference measure

In this section, we detail how we parametrize *in practice* the reference measure $\mathcal{U}$ so it can be optimized like any other hyper-parameter of the Gaussian Process, by maximizing the log marginal likelihood of the observations. Importantly, this requires computing the derivatives of the log marginal likelihood by back-propagating through the GP kernel, and the Sinkhorn algorithm.

Algorithm 13 Learn Kernel parameters.

input $(P_i, y_i)_{1 \leq i \leq N}$: dataset of distributions.
input $\boldsymbol{\theta}_0 = (\mathbf{u}_0, \sigma_0, l_0)$: initial parameters.
repeat
    for all P_i **do**
        Solve regularized OT problem between $P_i, \mathbf{u}$.
        Compute Sinkhorn dual potential $g_{\mathbf{u}}^{P_i}$.
    end for
    Build Kernel $K_{ij} := \sigma^2 \exp - \frac{\|g_{\mathbf{u}}^{P_i} - g_{\mathbf{u}}^{P_j}\|^2}{2l^2}$.
    Compute log marginal likelihood $\mathcal{L}(\mathbf{u}, \sigma, l, K, y)$.
    Compute gradients $\nabla_{(\mathbf{u},\sigma,l)}\mathcal{L}$ with Auto-Diff.
    Perform one step of L-BFGS on $(\mathbf{u}, \sigma, l)$.
until convergence of $(\mathbf{u}, \sigma, l)$.
return optimal parameters (u_*, σ_*, l_*).

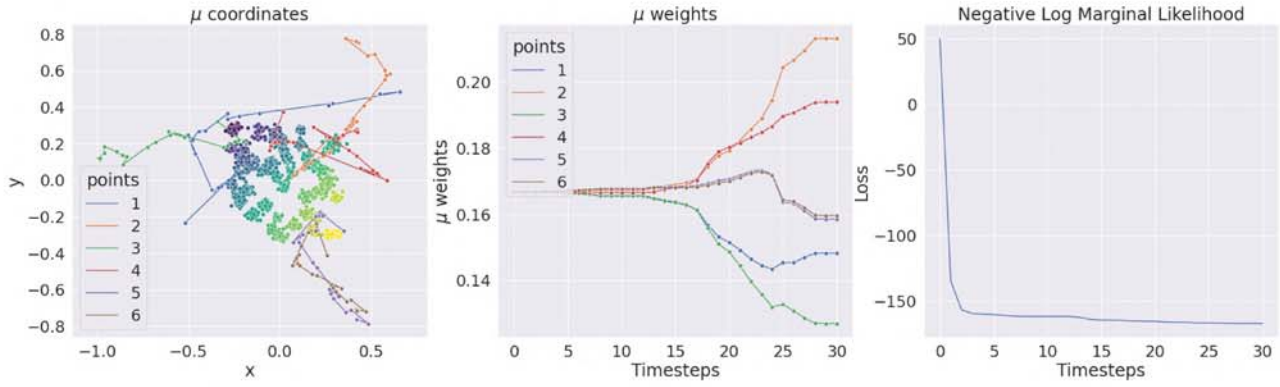


Figure C.1: **Toy example**. **Left**: 50 point clouds of the train set, with color scale depending on the random field Z. The trajectories of the points $\mathbf{x}_i$ of $\mathbf{u}$ are depicted in different colors. **Center**: evolution of the weights $\mathbf{w}$ of $\mathbf{u}$ during training. **Right**: evolution of the negative log marginal likelihood during training.

Task	embedding dimension	m	Ours	Bachoc et al. (2020)
Toy example	6	30	0.997	0.81

Table C.1: **Explained Variance Score (EVS) on the test set for regression tasks**, with train set of size $n = 50$ in dimension d=2. $|\mathbf{u}|$: dimension of the embedding. m: cloud size.

C.2.1 Parametrization of the Reference Measure $\mathcal{U}$

We choose a suitable machine representation for $\mathcal{U}$ as a weighted sum of Diracs:

$$\mathcal{U} = \sum_{i=1}^{|\mathbf{u}|} w_i \delta(\mathbf{x}_i) \text{ with } \sum_{i=1}^{|\mathbf{u}|} w_i = 1, w_i \geq 0, \mathbf{x}_i \in \mathbb{R}^d.$$

We denote by $|\mathbf{u}|$ the embedding dimension. In this form $\mathcal{U}$ is not absolutely continuous w.r.t. Lebesgue measure, but this is not an issue. The parameters $\mathbf{u}$ for $\mathcal{U}$ gather $w_1, \ldots, w_q, \mathbf{x}_1, \ldots, \mathbf{x}_q$. The procedure for the estimation of $\mathbf{u}, \boldsymbol{\theta}$ is sketched in Algorithm 13.

Other Numerical Aspects. For $\mathbf{u}$, the point coordinates are parameterized as $\mathbf{x} = S \tanh(\tilde{\mathbf{x}})$ with $S \in \mathbb{R}$ to ensure they remain bounded, the weights are parametrized as $\mathbf{w} = \text{softmax}(\tilde{\mathbf{w}})$ to ensure they represent a valid probability distribution. The dual variables $g^{\mathrm{P}}_{\mathbf{u}}$ computed at each time step during the optimization of $\mathbf{u}$ are cached to speed up Sinkhorn iterations: this strategy is reasonable since when $\mathbf{u}$ and $\mathbf{u}'$ are close then the dual variables $g^{\mathrm{P}}_{\mathbf{u}}$ and $g^{\mathrm{P}}_{\mathbf{u}'}$ are close too.

C.2.2 Gradient Computations

We will use the L-BFGS method for optimization Liu and Nocedal (1989). This requires the gradients of the likelihood function in regression and classification w.r.t. $\boldsymbol{\theta}$ and $\mathbf{u}$. The derivatives of relevant quantities w.r.t. $\boldsymbol{\theta}$ can be found in the literature, see for instance Rasmussen and Williams (2006). A specificity of $\mathbf{u}$ is that for some measures P, Q, we need to differentiate $\|g^{\mathrm{P}}_{\mathbf{u}} - g^{\mathrm{Q}}_{\mathbf{u}}\|_{L^2(\mathcal{U})}$ w.r.t. $\mathbf{u}$, that is we need to differentiate regularized OT plans. This is possible either by back-propagating through unrolled Sinkhorn iterations Genevay et al. (2018) or by using implicit differentiation Eisenberger et al. (2022). In practice, we noticed that, while being slower, unrolling Sinkhorn iterates was more stable numerically.

C.2.3 Computational Cost of u-Sinkhorn Kernels.

For another point cloud of size n, according to Altschuler et al. (2017); Dvurechensky et al. (2018) the time complexity of the Sinkhorn algorithm is $\mathcal{O}(\frac{n|\mathbf{u}|\log(n|\mathbf{u}|)}{\epsilon^2})$ to reach precision ϵ, while the complexity of the MMD kernel is $\mathcal{O}(n^2)$. It follows that for a reference measure with $|\mathbf{u}| \ll n$ the runtime cost of Sinkhorn $\mathbf{u}$-kernel becomes competitive. Runtimes against MMD are reported in Table C.3 (in the Appendix), with a speed-up of up to 100 for our method.

Once $\mathbf{u}$ is chosen, the embeddings $g^{\mathrm{P}}_{\mathbf{u}}$ can be pre-computed once for all for each point cloud $\mathrm{P}_1, \ldots, \mathrm{P}_n$ and used as a low dimension embedding of $\mathcal{P}(\Omega)$ into $\mathbb{R}^{|\mathbf{u}|}$. The distribution support $|\mathbf{u}|$ needs to be big enough to capture the similarities between the P_is up to the precision required by the task, but does not need to be bigger (see Section C.3.2).

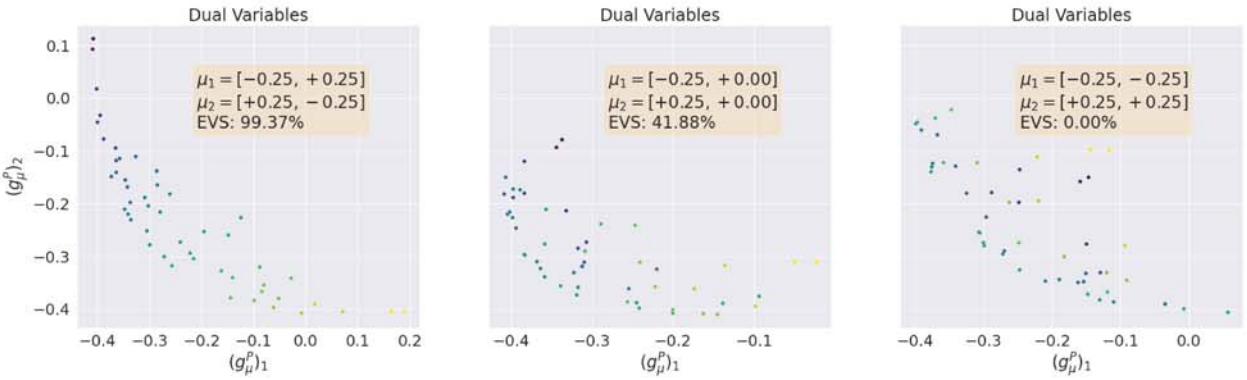


Figure C.2: **Role of u in quality of embeddings when** $|\mathbf{u}| = 2$ for the example of Section C.3.2. Each dot is the 2D embedding of a Gaussian where the color depends on the random field Z. **Left**: optimal choice for $\mathbf{u}$ that ensures the task can be solved. **Center**: sub-optimal choice for $\mathbf{u}$. **Right**: bad choice of $\mathbf{u}$ that prevents learning.

C.3 Experiments

In this section, we illustrate the algorithm on toy examples, and on 2015 US census data to predict the results of the 2016 election.

C.3.1 Implementation

For automatic support of autodifferentiation, we use the Jax framework (Bradbury et al., 2018) with the libraries GPJax (Pinder and Dodd, 2022) to implement GP regression, OTT-Jax (Cuturi et al., 2022) for differentiable Sinkhorn algorithm, and Jaxopt (Blondel et al., 2022) for optimization with Limited Memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS), which is an order 2 method, to enjoy faster convergence than order-1 methods such as Gradient Descent. The dominant cost of the algorithm is induced by the size of the support $\mathbf{u}$ and by the dimension of the points $\mathbf{x}_i \in \mathbb{R}^d$ since $(\sigma, l) \in \mathbb{R}^2$. The total dimension of search space is hence $nd + 2$. We select the optimal stepsize at each iteration with a *zoom line search* (Algorithm 3.6 of Nocedal and Wright (1999), pg. 59-61. Tries cubic, quadratic, and bisection methods of zooming). The computation of inverse covariance matrices is done efficiently using Cholesky decomposition (Press et al., 2007), which allows efficient computation of *matrix inverse-vector products* without materializing the inverse in memory. The computations are performed in *float32* arithmetic and take advantage of GPU for matrix operations, which are the bottleneck of the algorithm.

C.3.2 Regression on Toy Example

In this section we re-use the example introduced in Section 5.3 of Bachoc et al. (2020). We simulate 100 random two-dimensional isotropic Gaussian distributions. The means are sampled uniformly from $[-0.3, 0.3]^2$, and the variance uniformly from $[0.01^2, 0.02^2]$. The value of the random field induced by a Gaussian of means (m_1, m_2) and variance σ^2 is $Z = \frac{(m_1+0.5-(m_2+0.5)^2)}{1+\sigma}$. Gaussians are approximated by point clouds of size 30 sampled from the distribution. The dataset is splitted into train (50 clouds) and test (50 clouds). The $\mathbf{u}$-measure consists of 6 points on the ball of radius 0.5. Their position $\mathbf{x}_i$ and weight w_i are trained for 30 iterations jointly with kernel parameters. The results are highlighted in Figure C.1 and Table C.1. The role of $\mathbf{u}$ is investigated in Figure C.2 with $|\mathbf{u}| = 2$: the position of the $\mathbf{x}_i$'s makes the embedding more or less suitable for the downstream task,

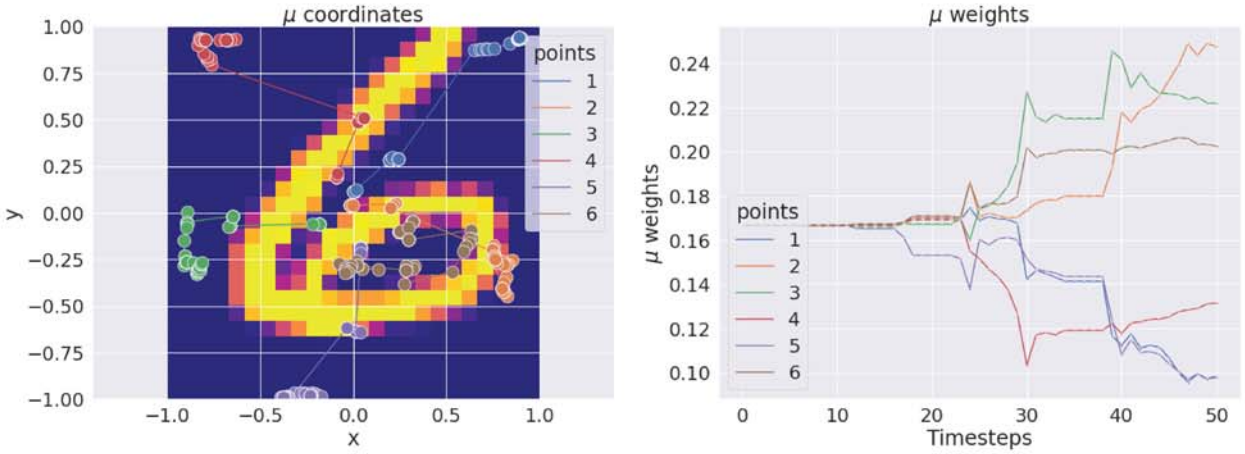


Figure C.3: **Optimization of u on Mnist "4" versus "6" task with $|\mathbf{u}| = 6$.** An image from the train set is displayed on the background to better grasp the scale of **u**.

as illustrated by the Explained Variance Score (EVS). The EVS of our method is higher than that of Bachoc et al. (2020).

C.3.3 Binary Classification on Mnist and Fashion-Mnist.

We perform binary classification on Mnist by learning to separate digits "4" and "6". The dataset consists of 200 train images, and 1000 test images. Each 28×28 images is centered crop to 24×24 to generate a cloud of size 576 matching pixel coordinates. The normalized pixel intensity is used as a weight in OT. The likelihood is modeled with Bernoulli distributions (not Gaussian, see the Appendix on GP classification), and the log marginal likelihood is maximized using maximum a posteriori (MAP) estimates. We tested different sizes for $|\mathbf{u}| \in [4, 5, 6]$. The training is depicted in Figure C.3. The experiment is repeated 10 time with random splits. It shows that Mnist images can be embedded in a space of small dimension that preserves most information about labels, achieving a compression rate of $R = \frac{|\mathbf{u}|}{584} \in [0.006, 0.013]$ tailored for the learning task. We also perform binary classification on Fashion-Mnist. We use a train set of size $n = 200$ in dimension d=2 with clouds of size $m = 24 \times 24 = 576$. We report the average over 25 runs.

Task	embedding dimension	Ours	RBF
"4" vs "6"	4	94.2 ± 1.2	✗
"4" vs "6"	5	95.5 ± 1.0	✗
"4" vs "6"	6	95.0 ± 0.6	98.8 ± 0.2
"shirt" vs "sandals"	12	99.5 ± 0.2	99.7 ± 0.2
"sneakers" vs "sandals"	12	88.6 ± 1.8	91.9 ± 1.2

It is approximately similar to the accuracy of a Radial Basis Function (RBF) kernel (also called squared exponential) applied to the "vectorized" images (see Section D.1.1). Remark that the RBF kernel cannot be applied to general point clouds, while our Sinkhorn kernel is designed for this. On Mnist and Fashion-Mnist, the MMD kernel could not provide comparable accuracy as Sinkhorn and RBF, due to its higher computational cost, see also Table C.3.

Dataset	u-Sinkhorn (ours)	RBF	Sliced Wasserstein (as reported by Kolouri et al. (2016))
UIUC Textures	87.2	87.3	$\mathbf{88 \pm 1}$
Mnist (1300 examples)	**92.50**	92.46	N/A

Table C.2: **Validation accuracy of C-SVM** with different kernels on GLCM embeddings of UIUC texture dataset Lazebnik et al. (2005), and 1300 example of Mnist (10 classes), with 5 folds cross-validation.

C.3.4 Texture Classification with C-SVM

We report here the results of classification with C-SVM on the University of Illinois Urbana Champaign (UIUC) texture dataset (Lazebnik et al., 2005), using the same protocol as Kolouri et al. (2016). Samples are shown in Figure C.4a. The dataset contains 25 different classes of texture on a total of 1000 images (only 40 images per class). We transform the images into two-dimensional probability distributions by computing the gray-level co-occurence matrices (GLCM) (Haralick et al., 1973). The Gray Level Co-occurrences Matrices (GLCM) are computed with the Scikit-image library (Van der Walt et al., 2014). The images are illustrated in Figure C.4b. The parameter γ of the SVM is obtained by following the "scale" policy of Scikit-learn library, applied on normalized features. We apply a grid search in logspace on the parameter C of SVM, ranging from 10^{-1} to 10^3. The optimal parameter is selected by the highest average accuracy in 5-fold cross-validation.

Remark C.2. Why PSD kernels

Solving C-SVM is a quadratic programming problem. The PSD property of the kernel ensures that the problem is convex, and this convexity guarantees that the algorithm will converge to a global minimum. This is one of the major advantages of SVM over non-convex methods like neural networks.

We compare the result against the RBF kernel applied to the raw (unprocessed) pixels. The results are reported in Table C.2. We see that the RBF kernel and our kernel have similar accuracies. We note that our implementation of the RBF kernel provides an higher accuracy for it than the one reported in Kolouri et al. (2016). Our kernel matches the performances of Kolouri et al. (2016) on the same experimental protocol, in Table C.2.

C.3.5 Runtime cost against MMD

We choose the MMD distance with RBF as inner kernel:

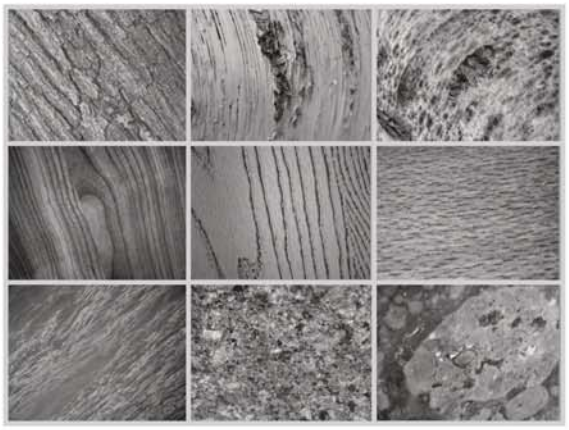
$$\mathrm{MMD}^2(\mathrm{P},\mathrm{Q}) = \mathbb{E}_{\mathrm{P}}(\mathrm{K}_{\mathrm{RBF}}(\mathbf{X},\mathbf{X}')) + \mathbb{E}_{\mathrm{Q}}(\mathrm{K}_{\mathrm{RBF}}(\mathbf{Y},\mathbf{Y}')) - 2\mathbb{E}_{\mathrm{P},\mathrm{Q}}(\mathrm{K}_{\mathrm{RBF}}(\mathbf{X},\mathbf{Y})), \tag{C.18}$$

with $\mathbf{X},\mathbf{X}' \sim \mathrm{P}$, $\mathbf{Y},\mathbf{Y}' \sim \mathrm{Q}$, with $\mathbf{X}$, $\mathbf{X}'$ $\mathbf{Y}$, $\mathbf{Y}'$ independent.

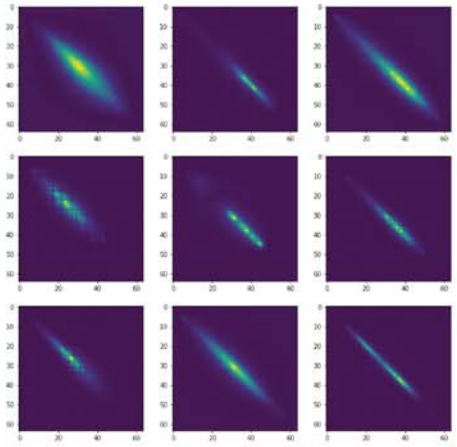
The MMD distance is turned into a kernel with an additional parameter σ:

$$K_{\mathrm{MMD}}(\mathrm{P},\mathrm{Q}) = \sigma^2 \exp\left(-\mathrm{MMD}^2(\mathrm{P},\mathrm{Q})\right). \tag{C.19}$$

The kernel in (C.19) is universal (see Theorem 2.2 of Christmann and Steinwart (2010) for example).



(a) **Random samples from the UIUC texture dataset** (Lazebnik et al., 2005).



(b) **Gray Level Co-occurrences Matrix (GLCM)**, normalized into 2D distributions.

Number of clouds	Cloud size	Sinkhorn with $\lvert\mathbf{u}\rvert = 6$	Sinkhorn with $\lvert\mathbf{u}\rvert = 12$	MMD
$n = 50$	$m = 100$	0.009s ($\times$0.1)	0.001s ($\times$1)	0.001s
$n = 100$	$m = 100$	0.013s ($\times$0.4)	0.011s ($\times$0.5)	0.005s
$n = 100$	$m = 400$	0.007s ($\times$7.9)	0.021s ($\times$2.6)	0.055s
$n = 400$	$m = 400$	0.018s ($\times$37.9)	0.059s ($\times$11.6)	0.683s
$n = 400$	$m = 625$	0.026s ($\times$64.7)	0.088s ($\times$19.1)	1.681s
$n = 1000$	$m = 625$	0.064s ($\times$169)	0.147s ($\times$73.7)	10.834s
$n = 1000$	$m = 1000$	0.090s ($\times$157)	0.158s ($\times$89.9)	14.207s

Table C.3: **Runtime cost of Sinkhorn u-Kernel (ours) against MMD**. The cost reported corresponds to the overall process: computation of regularized OT plan and of the kernel for Sinkhorn $\mathbf{u}$, and computation of MMD distance for MMD. Clouds are in dimension $d = 2$.

For a fair comparison the Sinkhorn $\mathbf{u}$-Kernel and MMD kernel are benchmarked on the same hardware under '@jax.jit' compiled code to benefit from GPU acceleration. We report runtime results in Table C.3. The clouds all share the same coordinates (but not the same weights). The pairwise distances between points of the clouds are pre-computed to speed-up both MMD and Sinkhorn iterations. We notice that Sinkhorn takes advantage of pre-computing the low dimension embeddings in dimension $|\mathbf{u}| = 6$, independent of the cloud size. We chose $\epsilon = 10^{-2}$ as regularization parameter. The points $\mathbf{u}$ are sampled uniformly in the square $[0, 1]^2$, while points from the clouds P_i are a discretization of the square $[0, 1]^2$ with equally spaced coordinates. Our Sinkhorn $\mathbf{u}$-kernel shows a speed-up of up to a factor 100 compared to the MMD one.

C.4 Kernel ridge regression

In this section, we illustrate this novel kernel $\mathcal{U}$ in the context of kernel ridge regression, which is simpler than GP modeling. For a dataset $(x_1, x_2, \dots x_n)$ with targets $(y_1, y_2, \dots y_n)$ the prediction

$f(x)$ at a new measurement x takes the form:

$$f(P) = \sum_{i=1}^{n} \alpha_i K(P_i, P) \tag{C.20}$$

with the coefficients α given by:

$$\alpha = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{Y}, \tag{C.21}$$

where $(\mathbf{K})_{\mathbf{ij}} = \mathbf{K}_{\mathcal{U}}(\mathbf{P_i}, \mathbf{Q_j})$ is the kernel matrix of pairwise similarity measures. Nonetheless, this is similar to GP modeling in the sense that the *maximum a posteriori* (MAP) estimate of the GP is the same as ridge regression when the kernel is the same. In this new setting, however, we do not care about the likelihood or the variance estimates. The hyper-parameters are typically estimated with cross-validation instead of log marginal likelihood maximization.

C.4.1 Other kernels on distributions

In this section, we compare $K_{\mathcal{U}}$ with other kernels, the mean embedding, the Random Fourier Features (RFF) (Rahimi and Recht, 2007), and Sliced Wasserstein (Kolouri et al., 2016).

For the embedding based on the Sinkhorn distance. The embedding dimension is thus simply the number of points. These points of the reference measure $\mathcal{U}$ are randomly sampled. We set the entropic regularization to $\epsilon = 0.1$.

Consider finally the embedding based on the sliced Wasserstein distance as in Kolouri et al. (2016). Standard implementations of kernel methods for this embedding involve pairwise computations of one-dimensional optimal transport problems, with random directions. For instance, this is the case for the `Python` Optimal Transport (`POT`) toolbox (Flamary and Courty, 2017). Instead, we provide a `Numpy` implementation where we compute separately the embeddings $x_{\mu_i^N}$, with the definition $x_\mu(\theta, t) = F_{\mu_\theta}^{-1}(t)$, with $(\theta, t) \in \mathcal{S}^{d-1} \times [0, 1]$, see also Meunier et al. (2022a, Prop. 5), with $F_{\mu_\theta}^{-1}$ the c.d.f of measure μ_θ, and μ_θ is the measure of the random variable $X^T\theta$ when $X \sim \mu$, for θ some column vector. The numerically implemented embeddings are the values of $F_{\mu_\theta}^{-1}(t)$ on a discretization of $\mathcal{S}^{d-1} \times [0, 1]$. The embedding dimension is thus the size of the discretization, which plays the same role as the number of random directions discussed above. Once the embeddings are computed (with a cost linear in n), we compute the $n \times n$ covariance matrix of the kernel values at $(\mu_i^N)_{i=1}^n$. In Figure D.7, we check numerically the validity of our implementation, by comparing it with the numerical results from `POT`, for a toy example in dimension $d = 2$.

C.4.2 Convergence speed

In Bachoc et al. (2023b) new rates are given for kernel distribution regression with two-stage sampling. Under suitable assumptions, all kernel presented previously exhibit rates of the same form, that depends on some hyper-parameters a, b that are kernel-dependant. Therefore, on some tasks, some kernels converge faster than others. We study this on a synthetic task.

Exemple C.1. Regressing the number of modes of Gaussian mixtures

We illustrate the impact of n and N numerically, on the problem of regressing the number of modes of Gaussian mixtures. This use case was introduced by Oliva et al. (2014), and we consider the settings of Meunier et al. (2022a). The random $(\mu_i)_{i=1}^n$ are generated as follows. In dimension d, the number of modes p is uniformly sampled in $\{1, \dots, C\}$, where $C \in \mathbb{N}$ is a setting parameter. Then for each component $b \in \{1, \dots, p\}$ of the mixture, the

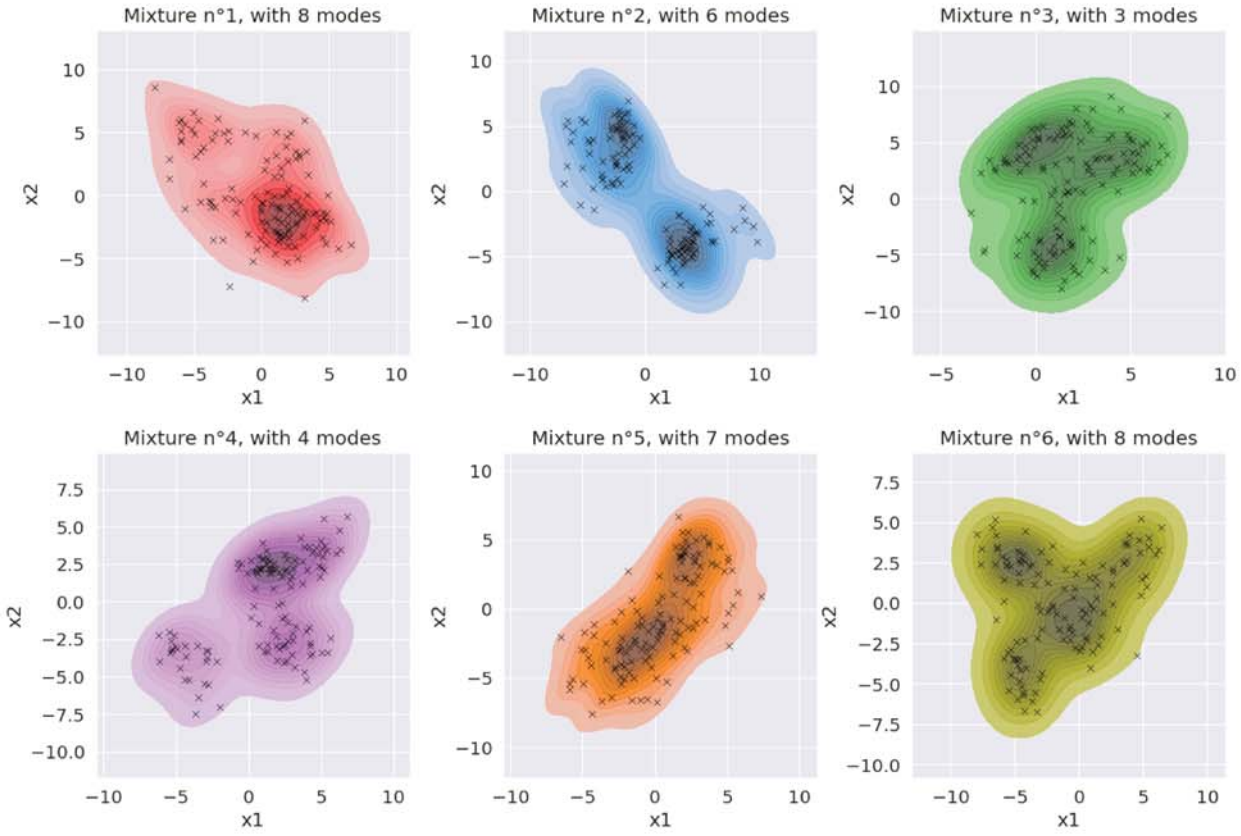


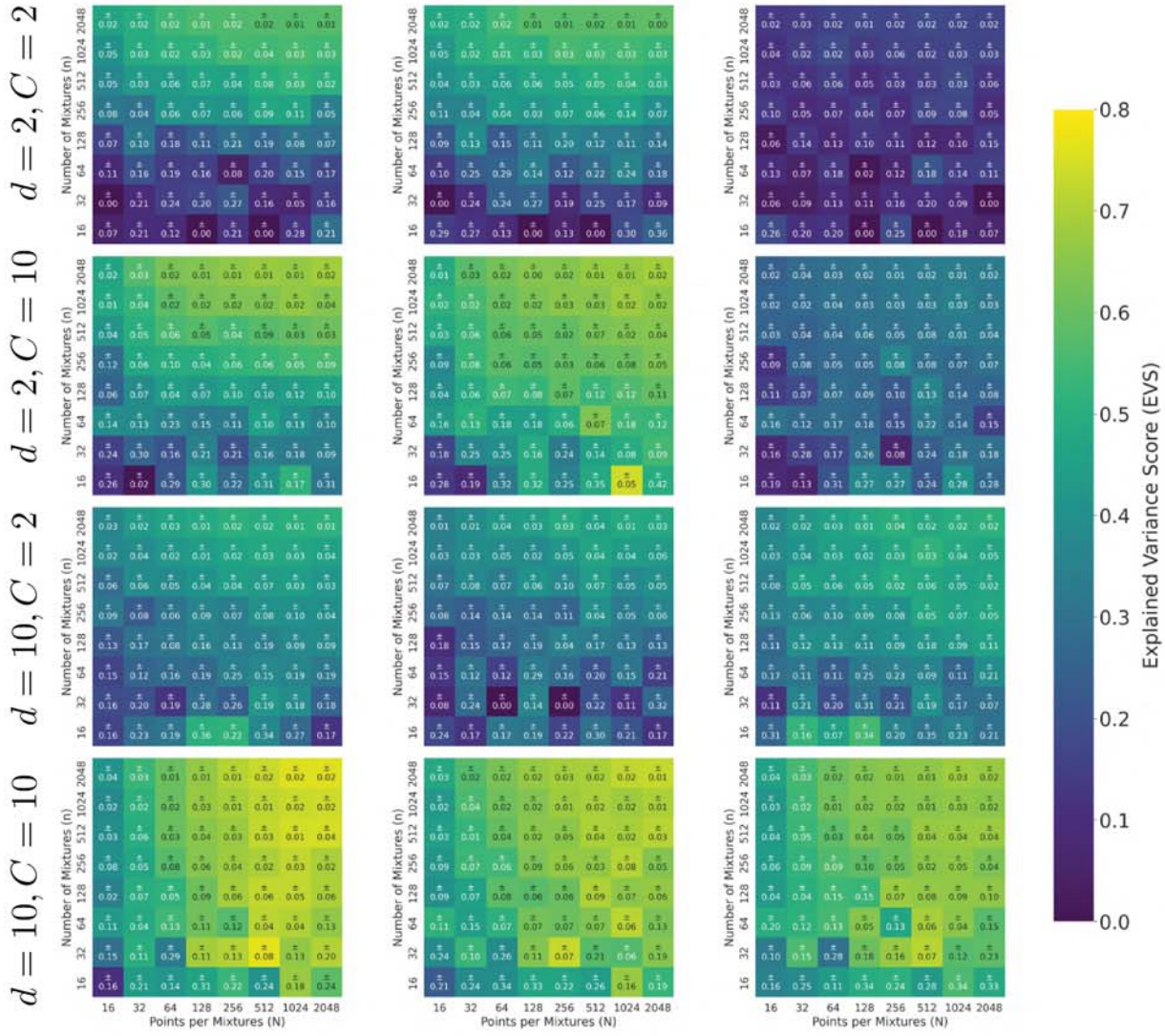
Figure C.5: **Examples of Gaussian mixture models used in the experiment of Section C.4.2**, in dimension $d = 2$ with at most $C = 10$ components per mixture.

mean vector is sampled as $m_b \sim \mathcal{U}([-5, 5]^d)$, and its associated covariance matrix is sampled as $\Sigma_b = a_b A_b A_b^\top + B_b$, where $a_b \sim \mathcal{U}([1, 4])$, A_b is a $d \times d$ matrix with entries sampled independently from $\mathcal{U}([-1, 1])$ and B_b is a diagonal matrix with entries sampled independently from $\mathcal{U}([0, 1])$. Therefore we set $\mu_i = \frac{1}{p} \sum_{b=1}^{p} \mathcal{N}(m_b, \Sigma_b)$ and $Y_i = p$ to define the i-th element of the dataset. We sample N points from each mixture μ_i. We illustrate the resulting dataset in Figure C.5.

We split each dataset into a train set containing 50% of the mixtures, and we evaluate the explained variance score on the test set composed of the remaining 50% mixtures.

The regularization parameter λ is selected in $\{10^{-2}, 10^{-1}, 1, 10, 10^2\}$ with cross validation on the train set. Furthermore, each "experiment" (that is each quintuple (C, d, n, N, λ), and there are 1 280 of them) is repeated 5 times, and the results are averaged, which adds up to 19 200 kernel ridge regressions in total. The averaged explained variance score as function of (n, N) is plotted in Figure C.6.

This experiment proves that the kernel plays the same role as the inductive bias of neural networks. Like neural networks, Reproducible Kernel Hilbert Spaces (RKHS), are universal approximators. However, like neural networks, they will require more or less samples to achieve meaningful generalization



(a) **Sinkhorn-$\mathcal{U}$.** (b) **Sliced Wasserstein.** (c) **Mean embedding.**

Figure C.6: **Explained variance score for different embeddings of distributions**, from the synthetic mode experiment described in section C.4.2, as function of the total number of distributions n, and the number of samples N per mixture. We plot the value of the explained variance score using the color, and the standard deviation with $\pm$ symbol. The dimension of the ambient space is denoted by d, and the maximum number of modes in the task is denoted by C.

Hilbertian embedding	Dim.	Hilbertian embedding runtime (↓ is better)	Ridge regression runtime (↓ is better)	Explained variance score in % (↑ is better)		Mean absolute error in % (↓ is better)	
				Democrat	Republican	Democrat	Republican
Constant baseline	0	00m00s	0.00s	0.	0.	12.4 ± 0.4	12.7 ± 0.4
Mean embedding (linear)	3899	**02m30s**	1.80s	27.4 ± 12	03.7 ± 6.2	10.0 ± 1.0	13.1 ± 5.0
Mean embedding (Fourier)	4096	09m33s	0.76s	$\mathbf{82.1 \pm 5.7}$	$\mathbf{83.1 \pm 2.3}$	$\mathbf{4.4 \pm 0.5}$	$\mathbf{5.0 \pm 0.3}$
Sliced-Wasserstein	1024	03m34s	0.32s	70.2 ± 5.6	72.74 ± 4.1	6.1 ± 0.5	6.2 ± 0.4
Sliced-Wasserstein	4096	03m44s	0.68s	75.9 ± 6.8	75.1 ± 3.3	5.3 ± 0.3	6.2 ± 0.3
Sinkhorn	**16**	26m49s	**0.16**s	50.6 ± 8.2	48.8 ± 5.1	7.9 ± 0.5	8.3 ± 0.5
Sinkhorn	32	28m27s	**0.16**s	67.1 ± 4.6	66.0 ± 4.4	6.6 ± 0.3	6.9 ± 0.2
Sinkhorn	64	30m42s	0.23s	61.7 ± 3.0	59.8 ± 4.0	7.1 ± 0.3	7.6 ± 0.3

Table C.4: **We perform distribution regression to predict percentages of Democrat and Republican vote for the 2016 US presidential election**, from socio-economics features extracted from 2015 US census data. We report the explained variance score and the mean absolute error over the test set, averaged over 5 random train/test splits of sizes 80% / 20% respectively. We also report the runtime required to compute the Hilbertian embeddings and to perform ridge regression on the embeddings. Best scores per column are in bold font.

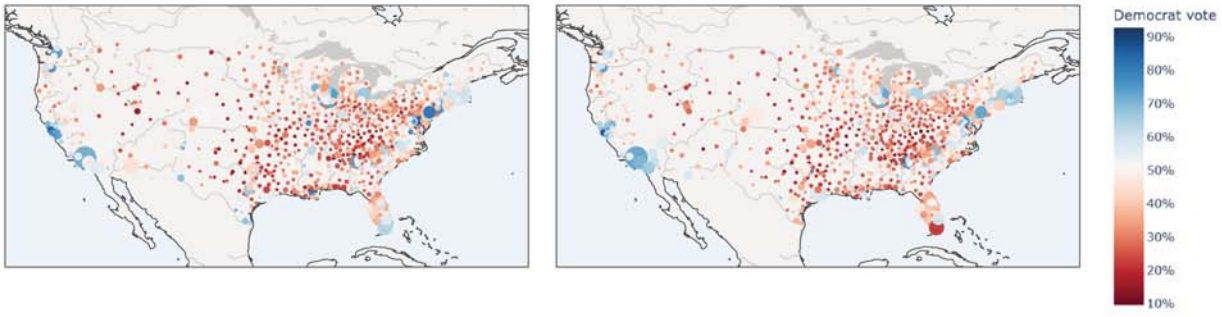
C.4.3 Ecological regression

We showcase an application of distribution regression to ecological inference, inspired by the seminal work of Flaxman et al. (2015).

Exemple C.2. Predicting results of 2016 US presidential election.

We use 2015 US census data, covering 2 490 616 individuals $X_{i,j}$ (0.75% of the 2015 US population), and totalling 3 899 features each (with one-hot encoding of categorical ones), covering characteristics like gender, age, race, occupation, schooling degree or personal income. This yields a fine-grained dataset of US demographics over $n = 979$ regions μ_i, spanning the 50 American states (20 regions per state on average, and $N = 2500$ individuals $X_{i,j} \sim \mu_i$ per region on average). We consider three targets $Y_i \in [0, 1]$ from the results of the 2016 presidential election: percentages of Republican vote, Democrat vote, and "Other" vote. We perform distribution regression by adapting the `pummeler` package of Flaxman et al. (2015, 2016) to compute the Hilbertian embeddings.

Table C.4 highlights global properties, and also specific benefits and drawbacks of each method. In Figure C.7, we provide a graphical example of successful distribution regression, for predicting the Democrat votes. We use the mean embedding with random Fourier features (having the best accuracy in Table C.4). It appears that the ecological inference is successful, as the structure of the Democrat vote is preserved between reality and prediction. In particular, the Democrat vote is well predicted in major cities of California and the Northeast. Among the rare exceptions to this accurate prediction, one can notice the extreme south of Florida, where the Democrat vote is strongly under-estimated. Indeed, at the time of the survey, Florida was a Democrat state. Interestingly, Florida is considered a “swing state”, and during mid-term elections in 2022, the Republican vote came on top.



(a) Democrat vote in the 2016 US presidential election.

(b) Distribution regression from socio-economics features: 4.4% of mean error.

Figure C.7: **Predicted and actual Democrat vote**, in the 2016 US presidential election, in each of the 975 regions (Hawaii and Alaska excluded from the plot). The surface of the markers is proportional to the number of individuals in the 2015 US census data, totaling 2 490 616 individuals over the USA. The Democrat vote is successfully recovered from the socio-economics features.

C.5 Conclusion

In this chapter, we showed how regularized transport can be used to create kernels on distributions. The main idea is to transport every empirical measure P_i towards a common reference measure $\mathcal{U}$ and use the dual variable of the corresponding optimization problem as a finite-dimensional embedding of each distribution. Intuitively, each dual variable corresponds to a point in the support of $\mathcal{U}$, and measures how much mass from P_i is transported towards it. Therefore, $\mathcal{U}$ is an important hyper-parameter of the algorithm. By relying on Sinkhorn's algorithm, and by analyzing the likelihood function of the Gaussian process, we show that the algorithm can be made differentiable end-to-end. This allows to optimize $\mathcal{U}$ efficiently.

Appendix D

Differentiable Gaussian Processes on Distributions: annexes

D.1 Gaussian Process

D.1.1 Algorithmic details

All the experiments were run on the publicly available GPU Colab hardware.

The code can be found on the repository: `https://github.com/Algue-Rythme/SinkhornMuGP`.

Kernel

We use the kernel:

$$K(\mathrm{P},\mathrm{Q}) = \sigma^2 \exp\left(-\frac{\|g^{\mathrm{P}}_{\mathbf{u}} - g^{\mathrm{Q}}_{\mathbf{u}}\|^2}{2l^2}\right). \tag{D.1}$$

Here the parameters are the tuple $\boldsymbol{\theta} = (l, \sigma)$ where $l > 0$ is the length scale and $\sigma > 0$ the scalar variance. For simplicity we only train a Gaussian process with zero mean function. This does not prevent the GP to reach satisfying levels of EVS/accuracy as illustrated in the experiments. The RBF kernel uses a similar form:

$$K_{\mathrm{RBF}}(\mathbf{x},\mathbf{y}) = \sigma^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2l^2}\right). \tag{D.2}$$

Hence the RBF kernel can be applied to vectors (or matrices representing images), but cannot handle general point clouds.

D.1.2 Sinkhorn's algorithm

Sinkhorn's algorithm is an iterative algorithm that takes advantage of *approximately good* solutions. Hence, the dual variables are re-used from one optimization step to the other. Using small steps guarantees that the initialization is not far away from the optimum. It allows the algorithm to benefit from a significant speed-up.

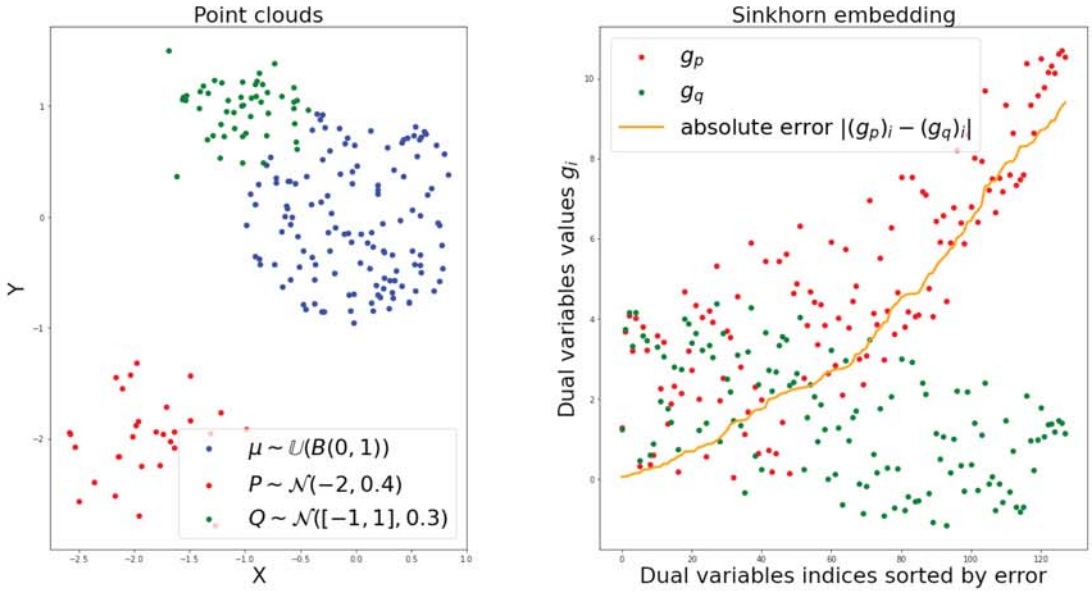


Figure D.1: **Vizualization of dual variables g^{P} and g^{Q}.** For P we sample 30 points from $\mathcal{N}([-2,-2], 0.4)$ and for Q we sample 50 points from $\mathcal{N}([-1,1], 0.3)$. We chose for $\mathbf{u}$ a finite sample of size 120 from the unit ball $\mathbb{B}(\mathbf{0}, \mathbf{1})$.

D.1.3 Dependance on reference measure

Visualizing dual variables In Figure D.1 we introduce an example with two distributions P and Q obtained by taking finite samples from isotropic Gaussians. For P we sample 30 points from $\mathcal{N}([-2,-2], 0.4)$ and for Q we sample 50 points from $\mathcal{N}([-1,1], 0.3)$. We choose for $\mathbf{u}$ a finite sample of size 120 from the unit ball $\mathbb{B}(\mathbf{0}, 1)$. We plot both the distributions and the values taken by g^{P} and g^{Q} respectively, by sorting dual variables arbitrarily by increasing error of $|g_i^{\mathrm{P}} - g_i^{\mathrm{Q}}|$.

Dependence on dimension

In figure D.2 we illustrate the dependence of the dimension of ambient space d on the convergence speed. The reference measure $\mathbf{u}$ is chosen to be 128 points sampled uniformly in unit ball. The task consists of $m \sim \mathbb{U}([100; 200])$ sampled at random from a Gaussian whose center is also sampled uniformly at random in range $\mu_i \sim \mathbb{U}([-10, 10])$. The regression task is the prediction of the mean $Y_i = \mu_i$ of each Gaussian from the finite sample. We use a Support Vector Regression machine (SVR) to perform the task. We report the Normalized mean Square Error by dividing by the dimension d to allow fair comparison on the same scale. We see that convergence speed is similar.

Toy dataset All clouds are centered and rescaled so the overall dataset (obtained by merging all clouds) has zero mean and unit variance across all dimensions. We study a discretization of $\mathbf{u}$ in the experiment of Section C.3.2. We choose $\mathbf{u}$ to be a discretization of the input space $[0,1]^2$ as a 50×50 grid. The density is chosen uniform over this discretization of 2500 points. Hence each regularized optimal transportation plan is between a Gaussian and an uniform measure over the square $[0,1]^2$. In this case the dual variable $g_{\mathbf{u}}^{\mathrm{P}}$ can be visualized as an image in definition 50×50. For 20 train examples, we plot the images $g_{\mathbf{u}}^{\mathrm{P}}$ in Figure D.3. We see that all those images appear "blurry", which shows the role of regularization in OT. Moreover those images seem to correspond to

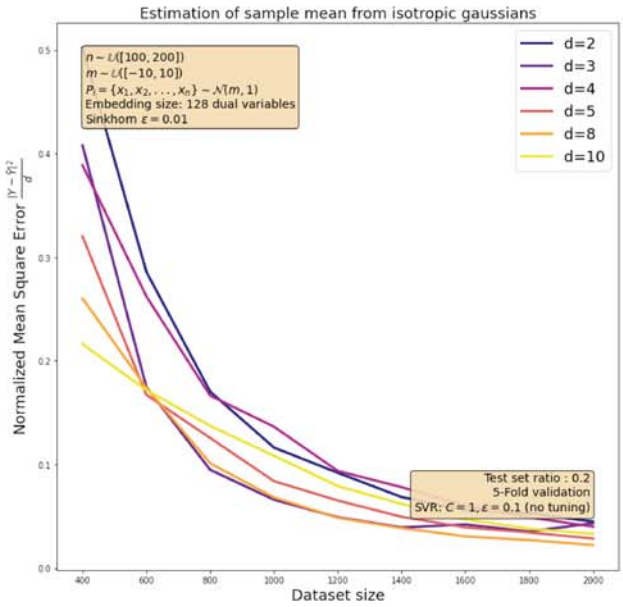


Figure D.2: **Normalized Mean Square Error as function of dimension** d and train set size n for the synthetic task of predicting the mean $\mu_i \sim \mathbb{U}([-10, 10])$ of a Gaussian from finite sample of size $m \in [100; 200]$.

a "blob" whose coordinates correspond to the ones of the clouds P_i. This figure helps to understand what the dual variables exactly look like in toy examples.

D.1.4 Mnist and Fashion-Mnist datasets

For RBF kernel, the images are normalized so that the pixel intensity lies in $[0, 1]$ range. Figure D.4 illustrates the evolution of $\mathbf{x}_i$'s and w_i's for $\mathbf{u}$ in the case of an image of shoe from Fashion-Mnist.

Sensivity to random affine transformations

In Figure D.5 we plot a set of Mnist images on which random affine transformations have been applied. We follow the protocol of Meunier et al. (2022a) and we sample a translation uniformly at random in range $[-6, 6]$ pixels, and a rotation uniformly at random in range $[-\frac{\pi}{3}, \frac{\pi}{3}]$ rads.

In Figure D.6 we study the influence of random affine transformations in dual variable space $g^{\mathrm{P}}_{\mathbf{u}}$, versus pixel space. In this experiment the reference measure $\mathbf{u}$ is chosen to have full support in dimension $28 \times 28 = 784$. The reference measure is chosen uniform on the pixel space. The images are processed as clouds of $28 \times 28 = 784$ pixels in dimension 2. The regularization factor is chosen to be $\epsilon = 10^{-2}$. We see that dual variables are less sensitive to translations than pixels. In the third row of Figure D.6 the dual variables are modified in a way that hints the direction and amplitude of translation, whereas in the fourth row the translation in pixel space has major consequences on the image and exhibits a huge Euclidean norm. This shows that μ Sinkhorn dual variables are better tailored to handle translates than conventional Euclidean metrics, thanks to the properties of OT in translations.

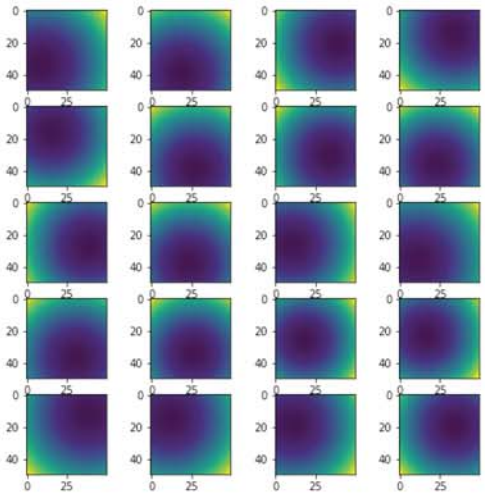


Figure D.3: **Plot of the dual variables $g_{\mathbf{u}}^{\mathrm{P}}$ for $5 \times 4 = 20$ distributions P_i from the toy example of Section C.3.2.** The position of the center of each "blob" (i.e. mean of the Gaussian) can be clearly seen by looking at the dual variables.

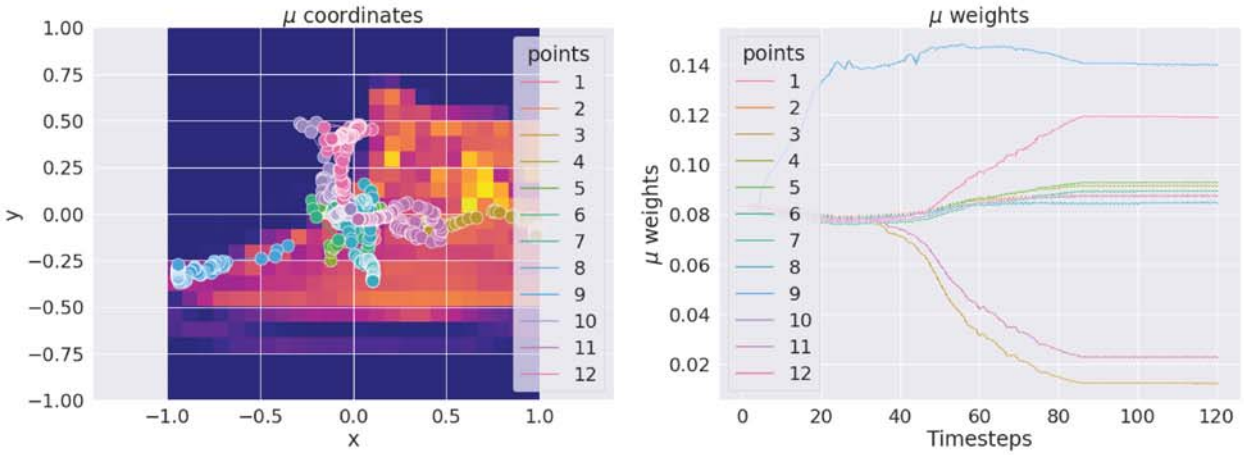


Figure D.4: **Evolution of $\mathbf{x}_i$'s and w_i's for $\mathbf{u}$ in the "sneakers" versus "sandals" task.**

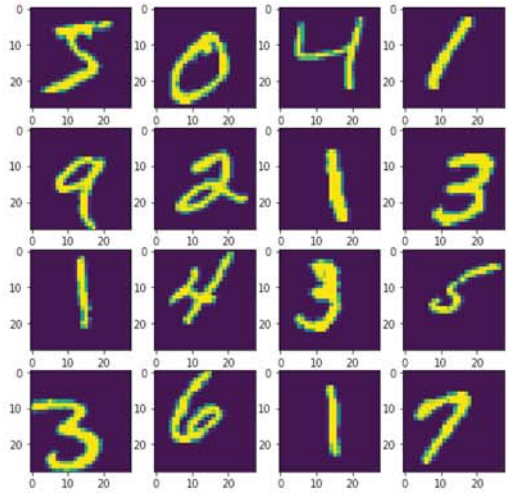


Figure D.5: **Mnist images with random affine transformations**: translation uniformly at random in range $[-6, 6]$ pixels.

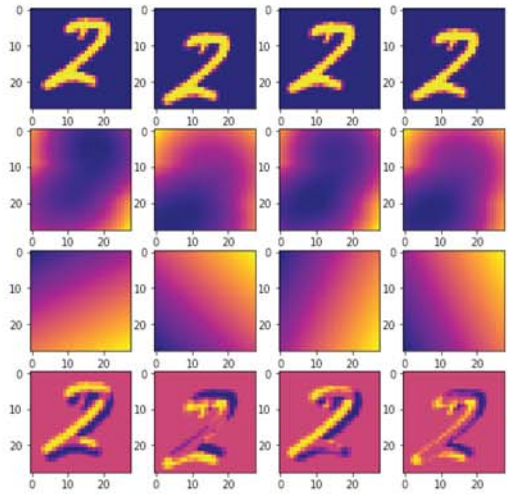


Figure D.6: **Visualization of translations on dual variables** for an Mnist image for $|\mathbf{u}| = 28 \times 28 = 784$. **Top row**: original image $\mathbf{x}$ with different affine transformations $\tilde{\mathbf{x}}$. **Second row**: Dual variables of translated images $g(\tilde{\mathbf{x}})$. **Third row**: pixel-wise difference between the dual variables of original (non modified) image and translated images $g(\mathbf{x}) - g(\tilde{\mathbf{x}})$. **Fourth row**: pixel-wise difference between original image and translated image $\mathbf{x} - \tilde{\mathbf{x}}$, in pixel space. We see that any translation has major impact in the pixel space, but only mild consequences in the dual variables space. Moreover the map $g(\mathbf{x}) - g(\tilde{\mathbf{x}})$ hints the nature of the translation, whereas $\mathbf{x} - \tilde{\mathbf{x}}$ is harder to interpret.

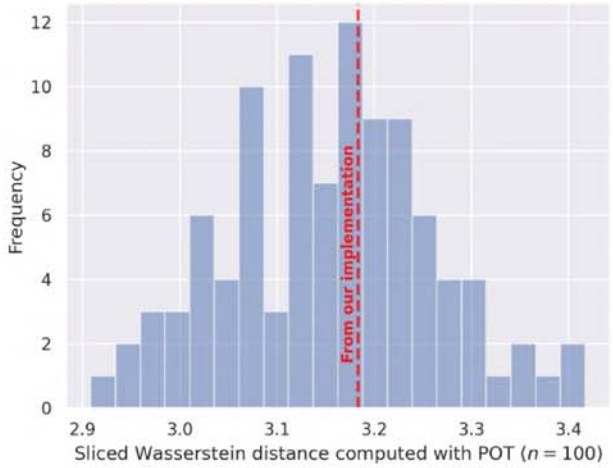


Figure D.7: **Comparison of implementations for the sliced Wasserstein distance.** Stochastic results from the `POT` toolbox (in blue), compared to the deterministic result from the Hilbertian mapping of our implementation (in red), for two samples of size 500 from Gaussian distributions $\mu = \mathcal{N}([0,0], \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix})$ and $\nu = \mathcal{N}([4,2], \begin{bmatrix} 2 & -0.8 \\ -0.8 & 1 \end{bmatrix})$. The results from `POT` are stochastic because of the random directions. For our deterministic result, we use a discretization of the half-circle with 25 directions of the form $(\frac{k}{25} - \frac{1}{2})\pi$ with $k = 1, \ldots, 25$, and a discretization of $[0,1]$ with 100 equidistant points. In both cases, we compute a finite-dimensional version of Sliced-Wasserstein kernel $\mathcal{SW}(\mu, \nu)$.

D.1.5 C-SVM results

Mnist C-SVM We choose a measure $\mathbf{u}$ with full support as in Section D.1.4. We select 1300 examples at random in the Mnist train set from all 10 classes, and we apply the protocol of Section C.3.4. The results of the best estimator found with 5 fold cross-validation are reported on an independent test set of size 1000 in Table C.2. Again, we have similar results as the RBF kernel.

D.2 Kernel ridge

D.2.1 Convergence speed

We give the details of Section C.4.2.

Recall that the explained variance is one minus the ratio of the empirical variance of the errors $\hat{Y}_i - Y_i$ on the test set, divided by the empirical variance of the data Y_i on the same test set.

We test the Mean Emdebbing, Sliced Wasserstein, and $K_{\mathcal{U}}$ on each different combination of values for $C \in \{2, 10\}$, $d \in \{2, 10\}$ and $\{n, N\} \subset \{16, 32, \ldots, 1024, 2048\}$.

For the mean embedding, we only consider the linear kernel. For the sliced Wasserstein embedding, we use a discretization of $\mathcal{S}^{d-1}$ with 10 random directions, and a discretization of $[0,1]$ with 10 equispaced points. For the embedding based on the Sinkhorn distance, we define the reference distribution $\mathcal{U}$ by sampling 100 points uniformly in the unit ball.

D.2.2 Ecological regression

For the Sinkhorn distance, we consider the support sizes 16, 32 and 64 for the reference distribution $\mathcal{U}$. For the generation of the points of $\mathcal{U}$, the numerical variables are sampled from the standard

normal distribution, while the categorical variables are sampled from a discrete distribution.

For the mean embedding, we consider the linear kernel k, for an embedding in dimension 3 899, and the embedding based on random Fourier features in dimension 4 096. For the sliced Wasserstein distance, we study the values 1,024 and 4,096 for the embedding dimensions, i.e. the number of discretization points. We find that directly regressing the probabilities $Y_i \in [0, 1]$ yields consistently better results than regressing their logarithms. Therefore we only report results involving the direct regression of these probabilities. We also standardize the features to improve the numerical stability of the computations. Finally, we enforce a default regularization parameter $\lambda = 10^{-3}$.

In Table C.4, we report the mean accuracies of the methods, averaged over 5 random train/set splits of sizes 80% (783 regions) / 20% (196 regions) respectively, together with the empirical variance with respect to the random seed. For interpretation purposes, we report the results achieved by the constant baseline prediction given by the empirical mean. We also report the runtime required to compute the embeddings from the raw US census data, and the runtime required to perform kernel ridge regression, given the embeddings.

The sliced Wasserstein embedding in dimension 1 024 is the fastest to compute (setting aside the linear mean embedding) and provides accuracies relatively close to the optimal one (with random Fourier features), for a significantly smaller embedding dimension (1 024 against 4 096). This is beneficial for dataset compression purposes. Hence, overall, the sliced Wasserstein embeddings provide an interesting tradeoff between runtime and final performance.

Overall, the accuracy and computation time increases with the embedding dimension. The mean embedding with the linear kernel yields the fastest embedding computation but also the lowest prediction accuracy. Hence, despite the high ambient dimension (3 899), a linear embedding is too restrictive. In contrast, the (non-linear) mean embedding with the random Fourier features yields the highest accuracy.

Finally, the Sinkhorn embeddings provide accuracies that are below those of the sliced Wasserstein ones and the mean embedding ones with Fourier features. On the other hand, the benefit of the Sinkhorn embeddings is that the embedding dimension is much smaller (a maximum of 64, against 1 024 to 4 096 for the other ones). Again, this is beneficial for dataset compression purposes and opens a non-linear dimension reduction prospect.

In Figure C.7, we use the mean embedding with random Fourier features (having the best accuracy in Table C.4). We split the dataset into 5 disjoint folds of sizes 195 or 196 each, we fit a kernel ridge regressor on four of the splits, and display its predictions on the fifth one, thus preventing overfitting.