Leveraging (Physical) Randomness in Machine Learning Algorithms

Ruben Ohana

SORBONNE UNIVERSITÉ

THÈSE DE DOCTORAT
DE SORBONNE UNIVERSITÉ

Préparée au Laboratoire de Physique
de l'Ecole Normale Supérieure

Leveraging (Physical) Randomness in Machine Learning Algorithms

Soutenue par
Ruben OHANA
Le 16 Septembre 2022

École doctorale n°564
Physique en Île-de-France

Spécialité
Machine learning

Préparée à
Ecole Normale Supérieure

Composition du jury :

Joan BRUNA New York University	*Rapporteur*
Rémi GRIBONVAL Inria & Ecole Normale Supérieure de Lyon	*Rapporteur*
Claire BOYER Sorbonne Université	*Examinatrice*
Julie GROLLIER CNRS & Thalès	*Présidente du jury*
Liva RALAIVOLA Criteo AI Lab	*Examinateur*
Alessandro RUDI Inria & Ecole Normale Supérieure	*Invité*
Laurent DAUDET LightOn	*Invité*
Florent KRZAKALA Ecole Polytechnique Fédérale de Lausanne	*Directeur de thèse*

SORBONNE UNIVERSITÉ ENS | PSL LPENS LABORATOIRE DE PHYSIQUE DE L'ÉCOLE NORMALE SUPÉRIEURE cnrs îledeFrance LightOn We bring Light to AI

Acknowledgments

I would like to start by thanking the members of my PhD jury Joan, Rémi, Claire, Julie and Liva ; it is an honor to have you in my jury and to present my work to you. Thanks to my supervisors Florent, Laurent and Alessandro as your presence was essential to my scientific development, I learned many things from you, from machine learning, to physics and mathematics.
Many thanks to the LightOn team, Julien (we are in this together !), Iacopo, Alessandro, Daniel, Baptiste, Igor, Adrien, Kilian, and all the others from the hardware and business teams. My PhD would not have been the same without all of you. I learned a lot about machine learning, but also on how to complain about the nearby Korean restaurant. I wish to LightOn a great success in the future !
I would like to continue by thanking the Sphinx team at ENS : Jonathan, Maria, Cédric, Marylou, Antoine, Benjamin, Alia, Sebastian, Stéphane, Francesca, Hugo. We had very good times together, I am pretty sure we will stay in contact !
I would also like to thank Liva and Alain for supervising me on the very last project of my PhD, and Kimia for working with me on this, I learned a lot and had fun during this project with you ! Of course, I shall not forget other co-authors, Jonas, Hamlet, Laurent and Maurizio whose expertise was essential on the projects we worked on together.
On a more personal note, I would like to thanks all my friends that were here for me during my whole studies. It took time but I made it ! Thanks for standing by my side. Special thanks to Pierre who bear with me during our tumultuous paths and to Arnaud who helped me with his english skills to correct the introduction of this thesis.
Je voudrais terminer en français, pour remercier mes parents, mon frère, mes grand-parents, mes tantes et tous mes proches. Sans vous tous à mes côtés, aller aussi loin aurait été impossible.
A mes parents, merci pour votre Amour et soutien inconditionnel. Si j'en suis ici aujourd'hui, c'est grâce à vous.

Acknowledgment of funding. This PhD was funded by the Paris Region PhD program of the Région Ile-de-France.

Foreword

Machine learning has become ubiquitous in everyday life. As a consequence, research and engineering in image recognition, text generation, recommender systems or data privacy – among the many sectors impacted – have experienced a deep transformation. Its broad adoption has also revolutionized numerous scientific fields, from physics to chemistry, computer science, mathematics, biology, bringing game-changing tools for data analysis and predictions.

Due to this important number of applications, it is now a priority to obtain more computationally and memory efficient algorithms. Indeed, machine learning architectures and algorithms becoming increasingly complex and large, a stark rise in computational demand is observed, challenging Moore's law (number of transistors in a dense integrated circuit doubling every two years). To palliate that, areas of research for efficient machine learning algorithms have emerged, focusing on improving three main aspects: the computational complexity (the scaling of the number of required operations with respect to the dataset size and dimension), the memory footprint, and their compatibility with the hardware (e.g. neural networks being "GPU-friendly").

In this thesis, we will leverage randomness at different stages within algorithms to improve their computational and memory efficiency. We will start by using a new optical hardware, with applications ranging from kernel methods to the training of deep neural networks. We will demonstrate how to perform optical random features that can scale better and be more energy efficient than GPUs. We will then show how a combination of the optical hardware and new training methods of deep neural networks, can lead to improving their security, by making them adversarially robust or differentially private, at a very small cost in natural accuracy contrary to standard defenses. We will also demonstrate how to accelerate and scale-up reservoir computing (a random recurrent neural network) by studying this method through the lens of random features. We will finish by studying sliced optimal transport by the means of the PAC-Bayesian framework, leading to a learning of the distribution of slices that is theoretically grounded.

More precisely, we will start by introducing the thesis in Chapter 1 and develop the technical background in Chapter 2.

In Chapter 3, we will explore random features for kernel approximation in the recurrent case (i.e. reservoir computing) and compute their *recurrent kernel* limit, corresponding to reservoirs with an infinite number of neurons. We will then introduce structured reservoir computing (by replacing the random matrices with structured transforms) to speed-up and reduce the memory footprint of reservoir computing. We will compare the regimes where recurrent kernels and structured reservoir computing are more computationally efficient than reservoir computing and prove their prediction capability on chaotic times-series prediction.

In Chapter 4, we will introduce *optical random features*, performed by an Optical Processing

Unit (OPU) and compute their kernel limit as a function of their exponent. We will demonstrate in which regime it is advantageous to use an OPU compared to a GPU for random matrix multiplication, as well as identify where the memory bottleneck of the GPU is. We will finish by showing in which regimes optical random features are more energy efficient than when performed on a GPU and show their prediction capabilities on standard benchmarks.

Motivated by the need of alternative training methods to backpropagation (of which the backward is sequential, i.e. updating the layers of the neural network one after the other), *Direct Feedback Alignment* (DFA) was introduced recently to unlock the parallel update of the weights thanks to a modification of the backpropagation update by incorporating a random projection of the error signal. This therefore allows bypassing non-differentiable layers. On the other side, the OPU is a physical process performing a random matrix multiplication with unknown weights, and is as a consequence non-differentiable. Building on these two key elements, we will show in Chapter 5, that placing the OPU before the classifier of a neural network, and updating the weights using DFA yields *adversarial robustness* by design. We will demonstrate the robustness on standard attacks for neural networks trained from scratch. When the neural network is already trained and robust (models taken from Robustbench), we will introduce ROPUST, a build-in block composed of the OPU, and placed just before the classifier. We show here that fine-tuning already robust models yields an increase of robustness at no cost of natural accuracy.

In Chapter 6, we will combine the OPU and DFA in another fashion. We will choose to perform the random matrix multiplication in the DFA update of the weights optically. By doing so, a small noise due to the measurement process of the camera of the OPU is added to the multiplication. We will show this yields a *differentially private* neural network, where, as expected, the amount of privacy is proportional to the variance of the noise. We derive the differential privacy parameters and show on standard benchmarks that our differentially private neural networks trained with DFA are much more robust to this noise compared to backpropagation-trained ones.

Relying on random projections, the Sliced-Wasserstein distance is a computationally efficient alternative to the Wasserstein distance. It consists in projecting into one dimension the distributions to compare using random projections, and exploiting the analytical formula of the Wasserstein distance for univariate distributions. On the other side, the PAC-Bayesian framework provides generalization bounds for classifiers sampled from a distribution. These bounds can be used to find an optimal distribution of parameters in a theoretically grounded fashion. In Chapter 7, we will combine the power of PAC-Bayesian bounds for optimizing the distribution of slices of Sliced-Wasserstein distances in a data-dependent way. This optimized distribution will benefit from generalization guarantees given by the PAC-Bayes bound and we will demonstrate this on numerical experiments including generative modelling.

Résumé

Dans cette thèse, nous tirerons parti de l'usage de l'aléatoire dans différents aspects de l'apprentissage automatique. Nous commençerons par montrer le lien entre le calcul par réservoir et les noyaux récurrents sous le prisme des caractéristiques aléatoires, et introduirons les transformées structurées afin d'améliorer la complexité computationnelle du calcul par réservoir. Par la suite, nous montrerons comment tirer parti de calculs optiques afin de mettre à l'échelle les caractéristiques aléatoires pour l'approximation de noyaux, à bas coût énergétique. Nous continuerons par montrer comment combiner le Processeur de Calcul Optique avec des méthodes d'entraînement alternatives à la rétropropagation du gradient tel que l'alignement par retour direct, afin de rendre adversariallement robuste des réseaux de neurones entraînés depuis le début ou d'augmenter la robustesse des défenses les plus robustes. Par ailleurs, nous entraînerons un réseau de neurones de façon optique et tirerons parti du bruit expérimental afin de démontrer comment cela induit une confidentialité différentielle. Nous finirons par utiliser les bornes PAC-Bayésiennes afin d'optimiser la distribution des projections aléatoires de la distance de Sliced-Wasserstein, tout en s'appuyant sur des fondations théoriques.

Mots clés : caractéristiques aléatoires, calcul optique, méthodes à noyau, calcul à réservoir, retour par alignement direct, robustesse adversarialle, confidentialité différentielle, transport optimal.

Abstract

In this thesis, we will leverage the use of randomness in multiple aspects of machine learning. We will start by showing the link between reservoir computing and recurrent kernels through the lens of random features, and introduce structured transforms to improve the computational complexity of reservoir computing. We will then show how optical computing can help scaling-up random features for kernel approximation, at a low energy cost. We will continue by showing how to combine the Optical Processing Unit with training methods alternative to backpropagation such as Direct Feedback Alignment, to make adversarially robust networks trained from the beginning, or improve the robustness of state-of-the-art defenses. We will also train optically a neural network and show how the experimental noise yields differential privacy. We will finish by using PAC-Bayesian bounds to optimize the distribution of random projections of Sliced-Wasserstein distances while being theoretically grounded.

Keywords : random features, optical computing, kernel methods, reservoir computing, direct feedback alignment, adversarial robustness, differential privacy, optimal transport.

Table of contents

Notations

$\mathbb{D}_\alpha(P\|\|Q)$	Renyi α-divergence between probability distributions P and Q
$\mathbf{0}$	d-dimensional vector with all components equal to 0
$\mathbf{1}$	d-dimensional vector with all components equal to 1
$\nabla_x f$	Gradient of the function f with respect to the variable x
$x \sim \mu$	x is a sample drawn from the probability distribution μ
$\Lambda(a, b)$	Laplace distribution with location parameter a and scale parameter b
δ_x	Dirac measure with mass on x
$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{H}}$	Inner product between $\boldsymbol{x}$ and $\boldsymbol{y}$ in the space $\mathcal{H}$
$\mathbb{E}_x$	Expectation over the random variable x
$\mathbb{N}$	Set of natural numbers
$\mathbb{P}$	Probability
$\mathbb{R}$	Set of real numbers
$\mathbb{S}^{d-1}$	Unit Sphere on $\mathbb{R}^d$, $\quad \mathbb{S}^{d-1} = \{\boldsymbol{\theta} \in \mathbb{R}^d \,:\, \|\boldsymbol{\theta}\| = 1\}$
$\mathbf{I}_d$	Identity matrix of size $d \times d$
$\mathcal{CN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Complex Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$\mathcal{P}(\mathcal{X})$	Set of probability distributions supported on $\mathcal{X}$
$\mathcal{P}_p(\mathcal{X})$	Set of probability distributions supported on $\mathcal{X}$ with finite p-th moment
$\mathrm{Det}(\mathbf{A})$	Determinant of the matrix $\mathbf{A}$
$\mathrm{Tr}(\mathbf{A})$	Trace of the matrix $\mathbf{A}$

Abbreviations

BP	Backpropagation
CDF	Cumulative Distribution Function
CPU	Central Processing Unit
DFA	Direct Feedback Alignment
DP	Differential Privacy
GPU	Graphical Processing Unit
i.i.d.	identically and independently distributed
KL	Kullback-Leibler (divergence)
ML	Machine Learning
OPU	Optical Processing Unit
ORF	Optical Random Feature
OT	Optimal Transport
PAC	Probably Approximately Correct
RC	Reservoir Computing
RF	Random feature
RK	Recurrent Kernel
SRC	Structured Reservoir Computing
SW	Sliced-Wasserstein (distance)
w.r.t.	with respect to

Chapter 1

Introduction

1.1 A short story of machine learning

Since the internet era began, numerical data has become increasingly present in our daily lives, and extracting information from it has raised many challenges. This motivated considerable efforts in research and engineering, where developing efficient and accurate algorithms, but also the hardware accelerating them, and appropriate building infrastructure (data-centers, power stations) became crucial. Due to the potential outcomes, the research on hardware components has grown at a impressive pace. The speed of technological progress in the last 50 years can be illustrated by Moore's law (see Figure 1.1) which observes that the number of transistors in integrated circuits (or grossly computing power) doubles every two years. However, at some point, the size of these transistors will reach the quantum limit, incorporating undesired effects in the transport of electrons [Powell, 2008]. Such a limit may slow down this incredible pace, and would require technological breakthroughs – certainly beyond silicon-based hardware – to maintain it.

On the other hand, along these more powerful chips, hardware-compatible algorithms were developed, and more specifically compatible with Graphical Processing Units (GPUs). Such algorithms include convolutions, backpropagation, matrix multiplication-based algorithms, or more generally algorithms composed of parallelizable computations. This compatibility drastically increased the pace of research and engineering in machine learning.

Nowadays, neural networks and backpropagation are used in the majority of large-scale applications of modern machine learning [LeCun, 2015], often combined with convolutional (for images) or attention layers (for images or text). This success is mainly due to the fact that they allow optimization of very complex and highly non-convex functions, reduced to matrix multiplication and gradient-descent based algorithms, with very accessible programming packages such as Pytorch [Paszke, 2019].

Modern machine learning breakthroughs. Most of the answers to highly complex modern problems usually require huge neural networks architectures with billion-scale parameters. The first widely publicized victory of AI/ML in games was Deep blue [Campbell, 2002], which beat

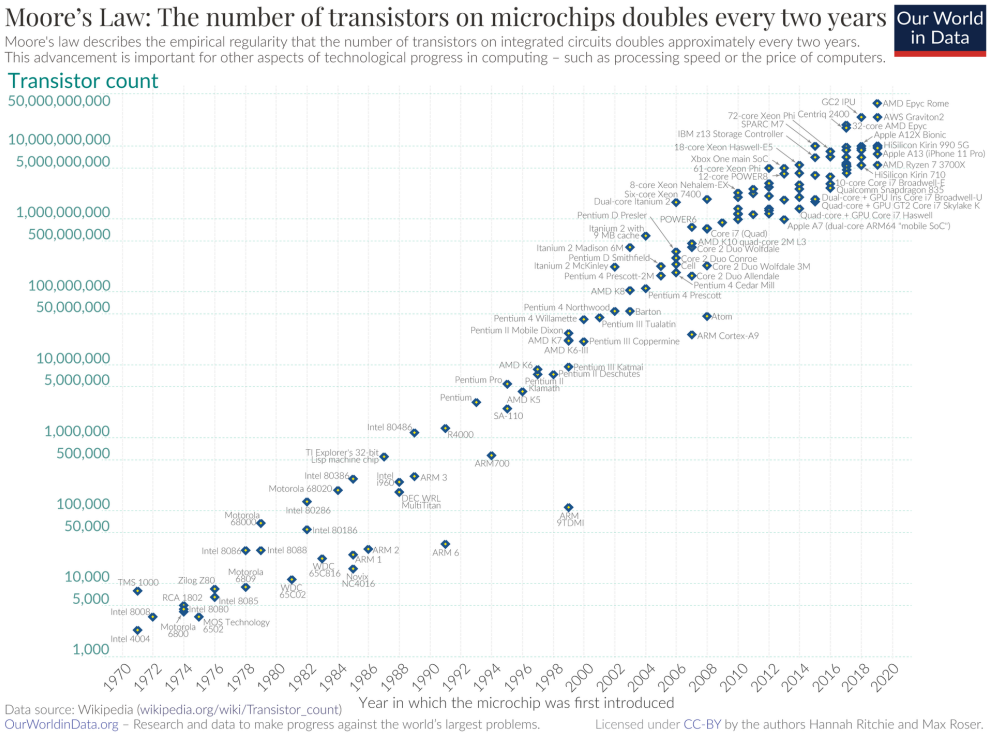


FIGURE 1.1 – Moore's law is the observation that the number of transistors on microchips roughly doubles every two years. This figure is a semi-log plot of the number of transistors versus the year of introduction of the processor. Source : Our world in data.

chess champions, followed by reinforcement learning-based algorithms beating experts in various games such as Go or Shogi [Silver, 2016 ; Silver, 2018]. Then the research focused on drastically improving accuracy on the protein folding problem [Jumper, 2021], the manipulation of plasma in Tokamaks [Degrave, 2022] and helped solving many problems for big physics instruments such as LIGO [George, 2018] or the CERN [Arpaia, 2021].

On another hand, Natural Language Processing went through a revolution during the past five years (as of mid-2022) with Bert [Devlin, 2018], GPT-2 [Radford, 2019] and GPT-3 [Brown, 2020], these models requiring billions of parameters and relying on transformers architectures. Images can also be created by entering a simple prompt with DALL-E [Ramesh, 2021 ; Ramesh, 2022]. All of these accomplishments mainly rely on big engineering issues to solve, but also on a huge computational power hardly available to public research laboratories. One of the expected benefits of more efficient algorithms and hardware is to make accessible these important topics to a wider range of researchers.

Hardware limitations. Due to this increasing demand in computational power, it is our role, as researchers, to develop more efficient algorithms, as well as novel hardware that can handle these huge computations, instead of just stacking an increasing number of GPUs (which relies on specialized engineering). In this thesis we will dedicate a three chapters focusing on optical computing for machine learning applications. Optical computing relies on light to perform computations and has the advantage to allow high-dimensional computations at a very low energy cost.

Security of systems. Due to the stark increase in the use of these models in critical applications, guaranteeing their security and privacy is now crucial. Indeed, as these systems handle sensitive information about users, any leakage could lead to dramatic consequences. That's why, in 2020, the US Bureau of Census used differential privacy when providing their data to avoid recovering information about users [Hawes, 2020]. Regarding private companies, Apple is using privacy as a selling feature for advertising their products (even if they don't communicate the level of privacy they provide). This had economical implications, as part of users information was blocked to companies using their devices, such as Meta, inducing a loss of 10 billion USD [1]. On another hand, protecting predictive systems against adversarial attacks – i.e. attacks that can fool the algorithm – is critical. For example, it can be used against self-driving cars which could lead to critical accidents [Eykholt, 2018].
In this thesis, we will modestly try to provide solutions to these challenges by leveraging randomness in machine learning models. As it can be incorporated at different steps within the learning algorithm, it will yield various desirable features. For instance, random reservoir computers allow the prediction on chaotic time-series at a very low computational cost compared to standard algorithms, and we will improve this cost at no loss in accuracy. When it is possible, randomness will be generated by an Optical Processing Unit, allowing the multiplication by a random matrix at scale unreachable by a GPU and at a lower energy consumption. We will demonstrate that this randomness generated optically can improve the security of a neural network, either being its robustness or privacy. We will also use Direct Feedback Alignment, an alternative training method to backpropagation that relies on a random projection of the final error at the end of the neural network. This method unlocks its backward pass, i.e. it allows the parallel update of weights. As this could lead to significant speed-ups in very large architectures, it is a method that we will study and take advantage of. We will finish by exploiting sliced optimal transport, a cheap but efficient alternative to standard optimal transport that relies on random projections, by providing some theoretical guarantees and improvements of this method.

1.2 Outline of the thesis

In Chapter 2, we will start by introducing mathematical and algorithmic concepts required to understand the themes studied thorough this thesis. As randomness will be studied under very different scopes, many technical elements will be introduced. We will also develop briefly works with minor contributions as they also study randomness and fit in the theme of this thesis.
In Chapter 3, we will develop methods to make more efficient Reservoir Computing (RC) approaches for chaotic times-series prediction. By identifying RC with recurrent random features, we will compute its infinite-width limit, namely a Recurrent Kernel, and compute convergence rates of the reservoir towards it. We will then introduce Structured Reservoir Computing, by leveraging the reduced computational cost of structured transforms to replace the random matrix of the reservoir and scale up Reservoir Computing approaches. We will then compute in which regime one has to replace Reservoir Computing by either Structured Reservoir Computing

1. https ://www.nytimes.com/2022/02/03/technology/apple-privacy-changes-meta.html

or Recurrent Kernels. We will finish by demonstrating the efficiency of our approaches on a multi-dimensional chaotic time-series. This chapter is based on [1] :

- [Dong, 2020] Jonathan Dong*, **Ruben Ohana***, Mushegh Rafayelyan, and Florent Krzakala. "Reservoir computing meets recurrent kernels and structured transforms." Advances in Neural Information Processing Systems 33 (NeurIPS 2020 - Oral Presentation) : 16785-16796.

In Chapter 4, we will perform the multiplication of the data by a random matrix, using a novel optical hardware – the Optical Processing Unit (OPU) – which uses light scattering through a diffusive medium. This will allow us to perform random features with the absolute value square element-wise non-linearity (due to the intensity measurement on the camera). We will compute the analytical kernel limit for arbitrary exponents of the feature map, making close links with polynomial kernels. Due to the optical nature of the operation, the allowed input and output dimensions of the random matrix can be of the order of millions. We will therefore compare speed of operation and power consumption with a GPU and show in which regimes the OPU outperforms it. We will finish by testing our optical random features on standard machine learning benchmarks to demonstrate their capabilities. This chapter is inspired from :

- [Ohana, 2020] **Ruben Ohana,** Jonas Wacker, Jonathan Dong, Sébastien Marmin, Florent Krzakala, Maurizio Filippone, and Laurent Daudet. "Kernel computations from large-scale random features obtained by optical processing units." International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020).

In Chapter 5, we will propose a new building block as an adversarial defense. This block is placed before the classifier of a neural network, and is composed of the Optical Processing Unit. Because the latter is non-differentiable, we will have to bypass it in the training step using an alternative training method to backpropagation, named Direct Feedback Alignment. We will demonstrate that our building block yields adversarial robustness against standard white-box, black-box and transfer attacks. We will then incorporate it in state-of-the-art robust models and just fine-tune the classifier. This will yield an increase in robustness, at no cost in natural accuracy. This chapter is inspired from :

- [Cappelli, 2021a] Alessandro Cappelli*, **Ruben Ohana***, Julien Launay, Laurent Meunier, Iacopo Poli, and Florent Krzakala. "Adversarial robustness by design through analog computing and synthetic gradients." International Conference on Acoustics, Speech and Signal Processing (ICASSP 2022).
- [Cappelli, 2021b] Alessandro Cappelli, Julien Launay, Laurent Meunier, **Ruben Ohana,** and Iacopo Poli. "Ropust : Improving robustness through fine-tuning with photonic processors and synthetic gradients." arXiv preprint arXiv :2108.04217 (2021).

In Chapter 6, we will use the Optical Processing Unit to perform the random matrix multiplication of Direct Feedback Alignment during the training step of a neural network. Because it is a physical measurement, we will assume the projection to be corrupted by Gaussian noise – due to the

1. The * denotes equal contribution.

photon shot-noise on the camera – which will make the neural network differentially privatewhere the level of privacy depends on the variance of the noise. We will compute the Differential Privacy parameters and then show on standard benchmarks that our methods yields differential privacy at a very small cost in accuracy compared to backpropagation trained networks.

- [Ohana, 2021] **Ruben Ohana***, Hamlet Medina*, Julien Launay*, Alessandro Cappelli, Iacopo Poli, Liva Ralaivola, and Alain Rakotomamonjy. "Photonic Differential Privacy with Direct Feedback Alignment." Advances in Neural Information Processing Systems 34 (NeurIPS 2021).

In Chapter 7, we will use the PAC-Bayesian framework to learn the distribution of random projections of Sliced-Wasserstein distances. Usually, this framework provides generalization bounds for the empirical risk of a classifier, and we will adapt it for Sliced-Wasserstein distances. The latter is a distance between probability distributions and depends on slices, usually sampled uniformly on the sphere. We will compute PAC-Bayesian bounds for these distances, which we will use to optimize the distribution of slices, to make these distances more discriminant. We will use a toy model to show a link with the distributional Sliced-Wasserstein distance and provide numerical experiments showing better generalization capabilities of our new distances. This Chapter is based on :

- [Ohana, 2022] **Ruben Ohana***, Kimia Nadjahi*, Alain Rakotomamonjy, Liva Ralaivola. "Shedding a PAC-Bayesian Light on Adaptive Sliced-Wasserstein distances." arXiv preprint arXiv :2206.03230 (2022).

1.3 Papers not included in this thesis

Theory of Direct Feedback Alignment. In this paper, we study the teacher-student setup, where we train the student network with (Direct) Feedback Alignment. We show that we first observe an alignment of the weights with the feedback matrix, followed by some loss in alignment to align with the teacher weights, going into the loss landscape of the network favoring alignment in both. The main results will be briefly developed in Section 2.3.1.1, which is based on :

- [Refinetti, 2021] Maria Refinetti*, Stéphane d'Ascoli*, **Ruben Ohana,** and Sebastian Goldt. "Align, then memorise : the dynamics of learning with feedback alignment." In International Conference on Machine Learning (ICML 2021), pp. 8925-8935. PMLR, 2021.

Complex-to-real random features. In this paper, we show that sampling complex random features for polynomial kernel approximation, then taking only the real part of the estimator, yields lower variance than real random features. The main results will be briefly developed in Section 2.2.4, which is based on :

- [Wacker, 2022a] Jonas Wacker, **Ruben Ohana,** and Maurizio Filippone. "Complex-to-Real Random Features for Polynomial Kernels." arXiv preprint arXiv :2202.02031 (2022).

1.4 Publications and personnal contributions

- [Ohana, 2020] **R. Ohana,** J. Wacker, J. Dong, S. Marmin, F. Krzakala, M. Filippone, L. Daudet. *Kernel computations from large-scale random features obtained by optical processing units.* IEEE ICASSP 2020.
  This works computes and studies the kernel limit of optical random features performed by Optical Processing Units. Personal contributions : development of the idea about the arbitrary exponent of the feature maps, computations of kernel limits and convergence rates, supervising experiments, writing of the paper.

- [Dong, 2020] J. Dong*, **R. Ohana***, M. Rafayelyan, F. Krzakala. *Reservoir Computing meets Recurrent Kernels and Structured Transforms.* NeurIPS 2020 (Oral Presentation).
  This work studies the kernel limit of Reservoir Computing and introduces Recurrent Kernels and Structured Reservoir Computing, improving the computational complexity. Personal contributions : development of the recurrent kernel limit idea, computation of the convergence rates, supervising numerical experiments, writing of the paper.

- [Cappelli, 2021a] A. Cappelli*, **R. Ohana***, J. Launay, L. Meunier, I. Poli, F. Krzakala. *Adversarial Robustness by Design through Analog Computing and Synthetic Gradients.* IEEE ICASSP 2022.
  This work studies the Optical Processing Unit as a adversarial defense against standard white-box, black-box and transfer attacks. Personal contributions : initial idea about the defense, preliminary white-box numerical experiments, black-box numerical results, writing of the paper.

- [Cappelli, 2021b] A. Cappelli, J. Launay, L. Meunier, **R. Ohana,** I. Poli. *ROPUST : Improving Robustness through Fine-tuning with Photonic Processors and Synthetic Gradients.* https://arxiv.org/abs/2108.04217.
  This paper studies the Optical Processing Unit as an adversarial defense for improving robustness of state-of-the-art robust models. Personal contributions : initial idea about the defense, analysis of experimental results.

- [Refinetti, 2021] M. Refinetti*, S. d'Ascoli*, **R. Ohana,** S. Goldt. *Align, then memorise : the dynamics of learning with feedback alignment.* ICML 2021.
  This paper studies the dynamics of learning with Direct Feedback Alignment algorithms. Personal contributions : original toy modelling for understanding alignment in DFA.

- [Ohana, 2021] **R. Ohana***, H. Ruiz*, J. Launay*, A. Cappelli, I. Poli, L. Ralaivola, A. Rakotomamonjy. *Photonic Differential Privacy with Direct Feedback Alignment.* NeurIPS 2021.
  This paper studies the differentially private mechanism induced by training a neural network with Direct Feedback Alignment performed by a Optical Processing Unit. Personal contributions : developing the linear OPU which triggered the initial ideas of this work, computation of the DP parameters of the model and of the main proposition, supervising numerical results, writing of the paper.

- [Ohana, 2022] **R. Ohana***, K. Nadjahi*, A. Rakotomamonjy, L. Ralaivola. *Shedding a PAC-Bayesian Light on Adaptive Sliced-Wasserstein Distances.* https://arxiv.org/abs/2206.03230.
  This paper uses the PAC-Bayesian framework to learn the distribution of slices of Sliced-Wasserstein distances while being theoretically grounded. Personal contributions : development of the scheme of proof and of the main theorem, struggling on $p = 2$ unbounded case and realizing it is too hard and may require new mathematical tools, development of the

link with the von-Mises Fisher distribution, numerical experiments on the generalization properties, writing part of the code, writing of the paper.

- [Wacker, 2022b] J. Wacker, **R. Ohana,** M. Filippone. *Complex-to-Real Random Features for Polynomial Kernels.* https://arxiv.org/abs/2202.02031.
  This paper introduces a new Complex-to-Real random feature for polynomial kernel approximation. Personal contributions : variance computations about the intuition behind these RFs, paper review.

- [Hesslow, 2021] D. Hesslow, A. Cappelli, I. Carron, L. Daudet, R. Lafargue, K. Müller, **R. Ohana,** G. Pariente, I. Poli. *Photonic co-processors in HPC : using LightOn OPUs for Randomized Numerical Linear Algebra.* Hot Chips 2021.
  This papers demonstrates that Randomized Numerical Linear Algebra can be performed using the OPU. Personal contribution : development of the linear OPU.

- [Brossollet, 2021] The LightOn team. *LightOn Optical Processing Unit : Scaling-up AI and HPC with a Non von Neumann co-processor.* https://arxiv.org/abs/2107.11814.
  This paper introduces the Optical Processing Unit. Personal contribution : development of new algorithms for the Optical Processing Unit thorough the thesis.

Patents

[Poli, 2021] I. Poli, J. Launay, K. Müller, G. Pariente, I. Carron, L. Daudet, **R. Ohana,** D. Hesslow. *Method and System for Machine Learning using Optical Data.* US 2021/0287079 A1.

Chapter 2

Technical Background

2.1 General machine learning principles

Machine learning can be divided in 3 main classes of problems : supervised learning (we have access to labels and data), unsupervised learning (unlabeled data) and reinforcement learning (an agent has a task or an environment to learn and solve). We will in this section develop the supervised learning setting, which is the one we will set ourselves in this thesis.
Let us give an informal example. If an user is building an application to automatically classify pictures of cats and dogs, the prediction model will be trained on already labeled data. Once it the training is performed, the model will be able to classify a new picture of a cat or a dog provided by the user. Behind this prediction is the process of learning a function (i.e. the model) mapping the new image to its class. This function learned on labeled data is in general highly complex and non-linear.
More formally, we will consider the data space to be $\mathcal{X} \subset \mathbb{R}^d$, and the label space to be $\mathcal{Y} \in \{-1, +1\}$ for (two class) classification or $\mathcal{Y} \subseteq \mathbb{R}$ for regression. We will assume that the training set $\mathcal{S} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ is sampled i.i.d. from a probability measure ρ on $\mathcal{X} \times \mathcal{Y}$.
The predictor f is chosen by a training algorithm from an *hypothesis class*, a set of functions from $\mathcal{X}$ to $\mathcal{Y}$, in order to minimize the error on the training set $\mathcal{S}$, called the *empirical risk* :

$$\hat{\mathcal{R}}(f, \ell, \mathcal{S}) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i), y_i), \tag{2.1}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a *loss function* which measures the quality of the prediction (or discrepancy) of $f(\boldsymbol{x})$ with respect to the true label y. Popular choices of ℓ include the ℓ_2-loss $\ell(y, f(\boldsymbol{x})) = (y - f(\boldsymbol{x}))^2$ for regression (used in linear regression and deep neural networks) or the hinge loss $\ell(y, f(\boldsymbol{x})) = \max(0, 1 - yf(\boldsymbol{x}))$ used in Support Vector Machines.
The goal of empirical risk minimization is to *generalize* on unseen data of the test set, by hoping that our function f minimizes the expected risk (or population risk) on the joint distribution ρ :

$$\mathcal{R}(f, \ell) = \mathbb{E}_{\boldsymbol{x},y}[\ell(f(\boldsymbol{x}), y)] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(f(\boldsymbol{x}), y) d\rho(\boldsymbol{x}, y), \tag{2.2}$$

which is expressed as the expectation over the joint distribution ρ. One can notice that the empirical risk is an unbiased estimator of the expected risk, i.e., $\mathcal{R}(f, \ell) = \mathbb{E}_{x,y}[\hat{\mathcal{R}}(f, \ell, \mathcal{S})]$.
One of the goals of researchers and machine learning engineers is to wisely chose the hypothesis class, the training method and the inductive bias, considering the type of data.

2.1.1 Linear regression

In the following, we will denote by $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^\top \in \mathbb{R}^{n \times d}$ the data matrix, i.e., the concatenation of the training points and $\boldsymbol{Y} \in \mathbb{R}^n$ the concatenation of the labels, i.e. $\boldsymbol{Y} = (y_1, \ldots, y_n)^\top$.
The simplest class of functions to learn is the class of linear functions, i.e. $f : \boldsymbol{x} \to \boldsymbol{x}^\top \boldsymbol{\beta}$, which is parametrized by $\boldsymbol{\beta} \in \mathbb{R}^d$. These parameters are usually learned by choosing the ℓ_2-loss (the method is called the *Ordinary least squares*). We want to find $\hat{\boldsymbol{\beta}}$ that minimizes the empirical risk defined in Equation 2.1 :

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \hat{\mathcal{R}}(\boldsymbol{\beta}, \mathcal{S}) = \frac{1}{n} \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 \tag{2.3}$$

$$= \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2. \tag{2.4}$$

This choice is motivated by some properties and assumptions such as linearity, homoscedasticity, independence of errors and of the data points.
Equation 2.4 can be solved analytically and has two equivalent solutions (if the columns of $\boldsymbol{X}$ are linearly independent) given by :

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y} = \boldsymbol{X}^\top (\boldsymbol{X}\boldsymbol{X}^\top)^{-1} \boldsymbol{Y}. \tag{2.5}$$

Once the above solution is computed on the train set, predicting the label $\boldsymbol{Y}_{\text{test}}$ of test points assembled in the matrix $\boldsymbol{X}_{\text{test}}$, we have to compute :

$$\boldsymbol{Y}_{\text{test}} = \boldsymbol{X}_{\text{test}} \hat{\boldsymbol{\beta}} \tag{2.6}$$

$$= \boldsymbol{X}_{\text{test}} (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{Y} = \boldsymbol{X}_{\text{test}} \boldsymbol{X}^\top (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{Y}. \tag{2.7}$$

2.1.2 Ridge regression

Ridge Regression (or Tikhonov regularization) is a method used for solving ill-posed problems which is usually the case in high-dimensional problems. It consists in restraining the parameters that are learned into high-dimensional balls of bounded radius. It consists in modifying Equation 2.4 by adding a regularization term in the ERM formulation :

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2, \tag{2.8}$$

with $\lambda \geq 0$. This is solved by :

$$\hat{\boldsymbol{\beta}} = \boldsymbol{X}^\top (\boldsymbol{X}\boldsymbol{X}^\top + \lambda \mathbf{I}_n)^{-1} \boldsymbol{Y} \tag{2.9}$$

$$= (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I}_d)^{-1} \boldsymbol{X}^\top \boldsymbol{Y}. \tag{2.10}$$

An interesting remark here is that these two solutions have different computational complexities, and one is to use in favor of the other depending the regime we are in. The inversion of an $n \times n$ matrix has a computational cost of $O(n^3)$. In knowledge of that, Equation 2.9 has a computational complexity of $O(n^3)$ coming from computing and inverting the $n \times n$ matrix $\boldsymbol{X}\boldsymbol{X}^\top$ and a memory cost of $O(n^2 + nd)$ for storing $\boldsymbol{X}\boldsymbol{X}^\top$ and $\boldsymbol{X}$. On the other hand, Equation 2.10 has a computational complexity of $O(d^3)$ for comuting and inverting $\boldsymbol{X}^\top \boldsymbol{X}$ and a memory cost of $O(d^2 + nd)$. Comparing these complexities, we can deduce that we should solve linear (ridge) regression using Equation 2.9 when $n < d$ and Equation 2.10 when $d < n$.

2.2 Kernel methods and random features

2.2.1 Kernel methods

Informally, kernel methods are non-parametric approaches to learning. Essentially, a kernel is a function measuring a (possibly non-linear) correlation between two points $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$. A specificity of kernels is that they can be expressed as the inner product of feature maps $\boldsymbol{\varphi} : \mathbb{R}^d \to \mathcal{H}$ in a possibly infinite-dimensional Hilbert space $\mathcal{H}$, i.e. $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{x}') \rangle_{\mathcal{H}}$. Kernel methods enable the use of linear methods in the non-linear feature space $\mathcal{H}$. The most famous kernel function is the Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}\right)$.
We need to introduce the following notions to introduce kernel methods more formally.

Definition 2.2.1 (Inner product [Schölkopf, 2002])**.** *Let $\mathcal{H}$ be a vector space over $\mathbb{R}$. A function $\langle \cdot, \cdot \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ is said to be an inner product on $\mathcal{H}$ if*

1. $\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle_{\mathcal{H}} = \alpha_1 \langle f_1, g \rangle_{\mathcal{H}} + \alpha_2 \langle f_2, g \rangle_{\mathcal{H}}$,
2. $\langle f, g \rangle_{\mathcal{H}} = \langle g, f \rangle_{\mathcal{H}}$,
3. $\langle f, f \rangle_{\mathcal{H}} \geq 0$ *and* $\langle f, f \rangle_{\mathcal{H}} = 0$ *if and only if* $f = 0$.

A norm is defined using the inner product as $\|f\|_{\mathcal{H}} := \sqrt{\langle f, f \rangle_{\mathcal{H}}}$.

Definition 2.2.2 (Hilbert space)**.** *A Hilbert space is a space on which a real or complex inner product space is defined, that is also a complete metric space (it contains the limits of all Cauchy sequences of it points) with respect to the distance function induced by the inner product.*

Definition 2.2.3 (Kernel function)**.** *Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel if there exists an $\mathbb{R}$-Hilbert space $\mathcal{H}$ and a map $\boldsymbol{\varphi} : \mathcal{X} \to \mathcal{H}$ such that $\forall\, \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$,*

$$k(\boldsymbol{x}, \boldsymbol{x}') := \langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{x}') \rangle_{\mathcal{H}}\,. \tag{2.11}$$

Definition 2.2.4 (Reproducing Kernel Hilbert Space (RKHS) [Scholkopf, 2002])**.** *Let $\mathcal{X}$ be a non-empty set and $\mathcal{H}$ a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Then $\mathcal{H}$ is called a reproducing*

kernel Hilbert space endowed with the dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ *if there exists a function* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *with the following properties,*

1. *k has the reproducing property, i.e.* $\langle f, k(\boldsymbol{x}, \cdot) \rangle_{\mathcal{H}} = f(\boldsymbol{x})\ \forall f \in \mathcal{H}$.
   In particular, $\langle k(\boldsymbol{x}, \cdot), k(\boldsymbol{x}', \cdot) \rangle_{\mathcal{H}} = k(\boldsymbol{x}, \boldsymbol{x}')$.
2. *k spans $\mathcal{H}$, i.e.* $\mathcal{H} = \overline{span\{k(\boldsymbol{x}, \cdot) | \boldsymbol{x} \in \mathcal{X}\}}$ *where* $\overline{X}$ *denotes the completion of the set* X.

Theorem 2.2.5 (Representer theorem [Scholkopf, 2002])**.** *Let* $\Omega : [0, \infty) \to \mathbb{R}$ *be a strictly monotonic increasing function,* $\mathcal{X}$ *be a set and* $\hat{\mathcal{R}}$ *defined in Equation 2.1. Consider a positive-definite positive kernel* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *with a corresponding RKHS* $\mathcal{H}$*. Then each minimizer* $f^* \in \mathcal{H}$ *of the regularized risk*

$$f^* = \arg\min_{f \in \mathcal{H}} \hat{\mathcal{R}}(f, \ell, \mathcal{S}) + \Omega(\|f\|_{\mathcal{H}}), \tag{2.12}$$

admits a representation of the following form,

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \cdot), \tag{2.13}$$

with $\forall i = 1, ..., n\ \alpha_i \in \mathbb{R}$.

The representer theorem is very useful in practice because it drastically reduces the computational cost of the problem. Indeed, in most machine learning problems, the minimization would be over infinite dimensional spaces $\mathcal{H}$ which is not possible to solve a on computer with finite memory and precision. Fortunately, the representer theorem allow us to reduce the possibly infinite dimensional problem on $\mathcal{H}$ into a n-dimensional minimization problem on the α_i's, which can be performed using standard machine learning algorithms.
This n-dimensional minimization problem is solvable through linear regression leading to a $O(n^3)$ computational complexity and a $O(n^2 + nd)$ memory complexity which scales badly when the number of points becomes high. For instance, let's take the MNIST [LeCun, 1990] database with a training set of 50000 points. A kernel matrix to be stored in memory scales quadratically with the number of points, would contain $50000^2 = 2.5 \times 10^9$ elements . This has to be multiplied by the precision of the storage, which in float32 (single-precision floating point format) requires 32 bits per element. Thus, this would require 80 GB of storage, which is not reasonable as it would not fit in the memory of modern GPUs.

2.2.2 Random features

Random features have been developed in [Rahimi, 2008] to overcome this issue. This celebrated technique introduces a random mapping $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^D$ such that the kernel is approximated in expectation. It was first developed for approximating the Gaussian kernel and relies on the following Bochner's theorem :

Theorem 2.2.6 (Bochner's Theorem [Bochner, 1933])**.** *Let* $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$*. A continuous kernel* $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x} - \boldsymbol{x}')$ *on* $\mathcal{X} \in \mathbb{R}^d$ *is positive definite if and only if* $k(\boldsymbol{x} - \boldsymbol{x}')$ *is the Fourier*

transform of a non-negative measure. Therefore, with proper scaling, $p(\boldsymbol{w})$ is a proper probability distribution :

$$k(\boldsymbol{x}-\boldsymbol{x}') = \int_{\mathcal{X}} e^{j\boldsymbol{w}^\top(\boldsymbol{x}-\boldsymbol{x}')}\, p(\boldsymbol{w})d\boldsymbol{w} = \mathbb{E}_{\boldsymbol{w}}[e^{j\boldsymbol{w}^\top\boldsymbol{x}}e^{j\boldsymbol{w}^\top\boldsymbol{x}'}], \tag{2.14}$$

where here $j^2 = -1$.

Therefore by sampling $\boldsymbol{w}$ from $p(\boldsymbol{w})$, the Fourier transform of the translation-invariant kernel, the kernel k can be approximated. This kernel, which is exactly the inner product of feature maps $\boldsymbol{\varphi}$ in an Hilbert space $\mathcal{H}$, is thus approximated in expectation by the inner product of random feature maps $\boldsymbol{\phi}$ in a chosen dimension D :

$$k(\boldsymbol{x},\boldsymbol{x}') = \langle \boldsymbol{\varphi}(\boldsymbol{x}), \boldsymbol{\varphi}(\boldsymbol{x}')\rangle_{\mathcal{H}} \approx \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}'), \tag{2.15}$$

with $\boldsymbol{\phi}(\boldsymbol{x}) = \frac{1}{\sqrt{D}}[f(\boldsymbol{w}_1^\top\boldsymbol{x}), ..., f(\boldsymbol{w}_D^\top\boldsymbol{x})]^\top \in \mathbb{R}^D$ and random vectors $\boldsymbol{w}_1, ..., \boldsymbol{w}_D \in \mathbb{R}^d$. Depending on the non-linearity f and the distribution of $\{\boldsymbol{w}_i\}_{i=1}^D$, we can approximate different kernel functions. For example, Random Fourier Features defined by :

$$\boldsymbol{\phi}(\boldsymbol{x}) = \frac{1}{\sqrt{D}}[\cos(\boldsymbol{w}_1^\top\boldsymbol{x}), ..., \cos(\boldsymbol{w}_D^\top\boldsymbol{x}), \sin(\boldsymbol{w}_1^\top\boldsymbol{x}), ..., \sin(\boldsymbol{w}_D^\top\boldsymbol{x})]^\top \in \mathbb{R}^{2D}, \tag{2.16}$$

approximate any translation-invariant (i.e. kernels such that $k(\boldsymbol{x},\boldsymbol{y}) = k(\boldsymbol{x}-\boldsymbol{y})$ and $k(0) = 1$). If $\boldsymbol{w}_1, ..., \boldsymbol{w}_D \sim \mathcal{N}(0, \sigma^{-2}\mathbf{I}_d)$, then the Gaussian kernel is approximated. The most popular non-linearities with their associated kernel are presented in Table 2.1 and can also be found in [Liao, 2018]. Note that we can also write Equation 2.16 in matrix form, i.e. $\boldsymbol{\phi}(\boldsymbol{x}) = \frac{1}{\sqrt{D}} f(\mathbf{W}\boldsymbol{x})$.

$f(\cdot)$	Associated kernel $k(\boldsymbol{x},\boldsymbol{x}')$
Erf$(\cdot)$	$\frac{2}{\pi}\arcsin\left(\frac{2\langle\boldsymbol{x},\boldsymbol{x}'\rangle}{\sqrt{(1+2\|\boldsymbol{x}\|^2)(1+2\|\boldsymbol{x}'\|^2)}}\right)$
RFFs : $[\cos(\cdot), \sin(\cdot)]$	$\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2}\right) = \exp\left(-\frac{\|\boldsymbol{x}\|^2+\|\boldsymbol{x}'\|^2-2\langle\boldsymbol{x},\boldsymbol{x}'\rangle}{2}\right)$
Sign$(\cdot)$	$\frac{2}{\pi}\arcsin\left(\frac{\langle\boldsymbol{x},\boldsymbol{x}'\rangle}{\|\boldsymbol{x}\|\|\boldsymbol{x}'\|}\right)$
Heaviside$(\cdot)$	$\frac{1}{2} - \frac{1}{2\pi}\arccos\left(\frac{\langle\boldsymbol{x},\boldsymbol{x}'\rangle}{\|\boldsymbol{x}\|\|\boldsymbol{x}'\|}\right)$
ReLU$(\cdot)$	$\frac{1}{2\pi}\left(\langle\boldsymbol{x},\boldsymbol{x}'\rangle\arccos(-\frac{\langle\boldsymbol{x},\boldsymbol{x}'\rangle}{\|\boldsymbol{x}\|\|\boldsymbol{x}'\|}) + \|\boldsymbol{x}\|\|\boldsymbol{x}'\|\sqrt{1-\left(\frac{\langle\boldsymbol{x},\boldsymbol{x}'\rangle}{\|\boldsymbol{x}\|\|\boldsymbol{x}'\|}\right)^2}\right)$

TABLE 2.1 – Table of point-wise non-linearities f and their approximated kernels. For any $\boldsymbol{x},\boldsymbol{x}' \in \mathbb{R}^d$ the kernel $k(\boldsymbol{x},\boldsymbol{x}')$ is the limit when D goes to infinity of $\langle\boldsymbol{\phi}(\boldsymbol{x}),\boldsymbol{\phi}(\boldsymbol{x}')\rangle = \langle\frac{1}{D}\langle f(\mathbf{W}\boldsymbol{x}), f(\mathbf{W}\boldsymbol{x}')\rangle$ with $\mathbf{W} \in \mathbb{R}^{D\times d}$ an i.i.d. standard random matrix.

Kernel Ridge Regression. A power of kernel methods is to combine them with Ridge Regression. This allows to performs the regression is a possibly infinite-dimensional space $\mathcal{H}$. Indeed, by transforming the data points using a feature map $\boldsymbol{\varphi} \in \mathcal{H}$, and looking at the solutions of linear

regression, one obtains :

$$\boldsymbol{Y}_{\text{test}} = \boldsymbol{\Psi}^*\boldsymbol{\Psi}^\top(\boldsymbol{\Psi}\boldsymbol{\Psi}^\top + \lambda\mathbf{I}_n)^{-1}Y = \boldsymbol{K}^*(\boldsymbol{K} + \lambda\mathbf{I}_n)^{-1}\boldsymbol{Y}, \tag{2.17}$$

where $\boldsymbol{\Psi}^* = (\boldsymbol{\varphi}(\boldsymbol{x}_1^*), \dots, \boldsymbol{\varphi}(\boldsymbol{x}_m^*))^\top$, $\boldsymbol{\Psi} = (\boldsymbol{\varphi}(\boldsymbol{x}_1), \dots, \boldsymbol{\varphi}(\boldsymbol{x}_n))^\top$, $\boldsymbol{K} = \boldsymbol{\Psi}\boldsymbol{\Psi}^\top \in \mathbb{R}^{n\times n}$, and $\boldsymbol{K}^* = \boldsymbol{\Psi}^*\boldsymbol{\Psi}^\top \in \mathbb{R}^{m\times n}$ and the stared elements are for the m elements of the test set. The elements of these matrices are $(\boldsymbol{K})_{ij} = \langle\boldsymbol{\varphi}(\boldsymbol{x}_i), \boldsymbol{\varphi}(\boldsymbol{x}_j)\rangle_{\mathcal{H}}$. As in linear regression, the computational complexity here is of $O(n^3)$ and one has to store the kernel matrix yielding a memory complexity of $O(n^2)$. On the other hand, the RF-based approximation turns a kernel-based model into a linear model with a new set of nonlinear features $\boldsymbol{\Phi} = (\boldsymbol{\phi}(\boldsymbol{x}_1), ..., \boldsymbol{\phi}(\boldsymbol{x}_n))^\top \in \mathbb{R}^{n\times D}$, i.e.

$$\boldsymbol{Y}_{\text{test}} = \boldsymbol{\Phi}^*(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\mathbf{I}_D)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{Y}. \tag{2.18}$$

This approximates the solution with the exact kernel function, but it yields a computation complexity of $O(D^3)$ to optimize the linear model and a memory complexity of $O(D^2)$. Therefore, when $D > n$, the user would therefore prefer to use standard kernel methods and when $D < n$, it is preferable on a complexity point of view, to use the random features method. Therefore, computing random features comes the main computational bottleneck.

	Kernel methods	Random features
Computational complexity	$O(n^3)$	$O(D^3)$
Memory complexity	$O(n^2)$	$O(D^2)$

TABLE 2.2 – Computational and memory complexities for Kernel Ridge regression using kernel methods or Random features. n is the number of training points, d their dimension and D the number of random features.

2.2.3 Orthogonal random features

The Walsh-Hadamard matrix is an orthogonal matrix ($\mathbf{H}_p^\top\mathbf{H}_p = \mathbf{I}_p$) defined recursively as :

$$H_0 = 1 \quad \text{and} \quad \mathbf{H}_p = \frac{1}{\sqrt{2}}\begin{bmatrix}\mathbf{H}_{p-1} & \mathbf{H}_{p-1}\\ \mathbf{H}_{p-1} & -\mathbf{H}_{p-1}\end{bmatrix}$$

It is therefore only defined in a square matrix form for dimensions $d = 2^p$ that are power of 2 (it can actually be defined for any arbitrary dimension d using simple tricks).
Thanks to the structure of this matrix, the computational complexity of its multiplication with a vector can be reduced from $O(n^2)$ to $O(n\log n)$ by using a divide-and-conquer strategy [Fino, 1976].
Fastfood [**Le, 2013**] was the first acceleration technique of random features based on Hadamard matrices. It consists in replacing the Gaussian random matrix with the following transform :

$$\mathbf{W}_{FF} = \frac{\sqrt{d}}{\sigma}\boldsymbol{SHG\Pi HB} \quad \in \mathbb{R}^{d\times d}, \tag{2.19}$$

where $\mathbf{\Pi} \in \{0,1\}^{d\times d}$ is a permutation matrix and $\mathbf{H}$ is the Walsh-Hadamard matrix developed above. $\boldsymbol{S}, \boldsymbol{G}$ and $\mathbf{B}$ are all diagonal random matrices where $\mathbf{B}$ has random $\{\pm 1\}$ entries, $\boldsymbol{G}$ has random Gaussian entries and $\boldsymbol{S}$ is a random scaling matrix. Here, σ is a parameter playing the equivalent role of a standard deviation of a Gaussian random variable. The coefficients of $\boldsymbol{S}, \boldsymbol{G}, \mathbf{B}$ are computed once and then stored, using $O(d)$ in memory because they are diagonal matrices.
Orthogonal Random features [Choromanski, 2017a] are a simpler transform and consist in replacing the Gaussian random matrix by a product of Hadamard matrices $\mathbf{H}$ and diagonal Rademacher distributed matrices $\mathbf{D}_i$, giving the following structured transform :

$$\mathbf{W}_{ORF} = \frac{\sqrt{d}}{\sigma} \prod_{i=1}^{s} \mathbf{H}\mathbf{D}_i \,, \tag{2.20}$$

where in standard applications, $s = 3$.
These transforms were first introduced to reduce the computational cost and memory footprint of random features approximation for kernel methods due to the cheap cost of the Hadamard matrix. However, their theoretical analysis and convergence properties toward their kernel limit is much more involved because the produced random features are not statistically independent. Therefore, standard convergence bounds such as Bernstein or Chernoff are not usable. However, recently, [Cherapanamjeri, 2022], succeeded to compute convergence rates when the activation is Lipschitz, justifying their empirical success. As it will be seen in Chapter 3, convergence rates of structured transforms are empirically the same as standard random features.

2.2.4 Minor contribution : Complex-to-Real (CtR) random features for polynomial kernel approximation

This section is an extract of [Wacker, 2022b].
A particularly important class of kernels are polynomial kernels $k(\boldsymbol{x}, \boldsymbol{y}) = (\gamma \boldsymbol{x}^\top \boldsymbol{y} + \nu)^p$ for some inputs $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, where $\gamma, \nu \geq 0$ and $p \in \mathbb{N}$. Their feature maps can be constructed explicitly and consist of interactions of the dimensions of $\boldsymbol{x}$ (or $\boldsymbol{y}$ respectively) up to order p, and are thus finite-dimensional. One could thus hope to avoid the scalability bottleneck of kernel methods by using explicit feature maps of the polynomial kernel in a parametric model. However, such hopes are dampened since the dimension of these feature maps scales as $O(d^p)$, which makes their construction practically infeasible as soon as d or p are moderately large.
In this section, we propose novel *Complex-to-Real (CtR)* polynomial random features that leverage intermediate complex projections to yield low-variance kernel estimates. The resulting features are real-valued and can be used inside any downstream task without requiring the model to handle complex data, e.g., as a drop-in replacement of the widely known random Fourier features [Rahimi, 2007].
We derive the variances of CtR random features using Gaussian and Rademacher projections in closed form and prove under which conditions they yield lower-variance kernel estimates than comparable random features in the literature that use only real-valued random projections.
More formally, recall that $p \in \mathbb{N}$ is the degree of the polynomial kernel and $D \in \mathbb{N}$ is the projection dimension of the polynomial random features. We generate $p \times D$ i.i.d. random weights

$\boldsymbol{w}_{i,\ell} \in \mathbb{C}^d$ satisfying $\mathbb{E}[\boldsymbol{w}_{i,\ell}\overline{\boldsymbol{w}_{i,\ell}}^\top] = \mathbf{I}_d$ for $i \in \{1, \dots, p\}, \ell \in \{1, \dots, D\}$, where $\mathbf{I}_d$ is the identity matrix of size d. Real-valued example distributions $p(\boldsymbol{w}_{i,\ell})$ that satisfy this property are the Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ as well as the Rademacher distribution, i.e., the elements of $\boldsymbol{w}_{i,\ell}$ are independently sampled from the uniform distribution over $\{1, -1\}$. Their complex-valued analogs are the complex Gaussian distribution $\mathcal{CN}(\mathbf{0}, \mathbf{I}_d)$ and the uniform distribution over $\{1, -1, \mathrm{i}, -\mathrm{i}\}$ with $\mathrm{i} := \sqrt{-1}$.
We define a polynomial random feature $\phi : \mathbb{R}^d \to \mathbb{C}^D$ as

$$\Phi(\boldsymbol{x}) := \frac{1}{\sqrt{D}} \left[\left(\prod_{i=1}^{p} \boldsymbol{w}_{i,1}^\top \boldsymbol{x}\right), \dots, \left(\prod_{i=1}^{p} \boldsymbol{w}_{i,D}^\top \boldsymbol{x}\right) \right]^\top, \tag{2.21}$$

and let $\hat{k}(\boldsymbol{x}, \boldsymbol{y}) := \Phi(\boldsymbol{x})^\top \overline{\Phi(\boldsymbol{y})} \in \mathbb{C}$ be the approximate kernel for two inputs $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, which is unbiased because

$$\mathbb{E}\left[\hat{k}(\boldsymbol{x}, \boldsymbol{y})\right] = \frac{1}{D} \sum_{\ell=1}^{D} \prod_{i=1}^{p} \boldsymbol{x}^\top \mathbb{E}[\boldsymbol{w}_{i,\ell}\overline{\boldsymbol{w}_{i,\ell}}^\top] \boldsymbol{y} = (\boldsymbol{x}^\top \boldsymbol{y})^p.$$

In the following, we show how to convert complex polynomial random features into real-valued CtR ones. Let $\Phi_\mathrm{C} : \mathbb{R}^d \to \mathbb{C}^D$ be a complex polynomial random feature and $\hat{k}_\mathrm{C}(\boldsymbol{x}, \boldsymbol{y}) = \Phi_\mathrm{C}(\boldsymbol{x})^\top \overline{\Phi_\mathrm{C}(\boldsymbol{y})}$ the approximate kernel defined previously, where we make the use of complex projections explicit through the subscript C. Let $\mathrm{Re}\{\cdot\}$ and $\mathrm{Im}\{\cdot\}$ denote the real and imaginary parts of a complex vector. $\hat{k}_\mathrm{C}(\boldsymbol{x}, \boldsymbol{y})$ is generally complex-valued and can hence be written as $\hat{k}_\mathrm{C}(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{Re}\{\hat{k}_\mathrm{C}(\boldsymbol{x}, \boldsymbol{y})\} + \mathrm{i} \cdot \mathrm{Im}\{\hat{k}_\mathrm{C}(\boldsymbol{x}, \boldsymbol{y})\}$.
As shown previously, $\hat{k}_\mathrm{C}(\boldsymbol{x}, \boldsymbol{y})$ is unbiased and thus $\mathbb{E}[\hat{k}_\mathrm{C}(\boldsymbol{x}, \boldsymbol{y})] = k(\boldsymbol{x}, \boldsymbol{y}) + 0 \cdot \mathrm{i}$. From this it follows that $\mathbb{E}[\mathrm{Re}\{\hat{k}_\mathrm{C}(\boldsymbol{x}, \boldsymbol{y})\}] = k(\boldsymbol{x}, \boldsymbol{y})$ and $\mathbb{E}[\mathrm{Im}\{\hat{k}_\mathrm{C}(\boldsymbol{x}, \boldsymbol{y})\}] = 0$ through the linearity of the expectation.
We define $\hat{k}_\mathrm{CtR}(\boldsymbol{x}, \boldsymbol{y}) := \mathrm{Re}\{\hat{k}_\mathrm{C}(\boldsymbol{x}, \boldsymbol{y})\}$ to be our novel kernel estimate which, by expanding the real part of $\Phi_\mathrm{C}(\boldsymbol{x})^\top \overline{\Phi_\mathrm{C}(\boldsymbol{y})}$, can be written as

$$\hat{k}_\mathrm{CtR}(\boldsymbol{x}, \boldsymbol{y}) := \mathrm{Re}\{\Phi_\mathrm{C}(\boldsymbol{x})\}^\top \mathrm{Re}\{\Phi_\mathrm{C}(\boldsymbol{y})\} + \mathrm{Im}\{\Phi_\mathrm{C}(\boldsymbol{x})\}^\top \mathrm{Im}\{\Phi_\mathrm{C}(\boldsymbol{y})\}.$$

Note that $\mathrm{Re}\{\Phi_\mathrm{C}(\boldsymbol{x})\}, \mathrm{Im}\{\Phi_\mathrm{C}(\boldsymbol{x})\} \in \mathbb{R}^D$, which allows us to define a $2D$-dimensional *real-valued* polynomial random feature

$$\Phi_\mathrm{CtR}(\boldsymbol{x}) := \left(\mathrm{Re}\{\Phi_\mathrm{C}(\boldsymbol{x})_1\}, \dots, \mathrm{Re}\{\Phi_\mathrm{C}(\boldsymbol{x})_D\}, \mathrm{Im}\{\Phi_\mathrm{C}(\boldsymbol{x})_1\}, \dots, \mathrm{Im}\{\Phi_\mathrm{C}(\boldsymbol{x})_D\}\right)^\top \in \mathbb{R}^{2D}, \tag{2.22}$$

for which we have

$$\Phi_\mathrm{CtR}(\boldsymbol{x})^\top \Phi_\mathrm{CtR}(\boldsymbol{y}) = \mathrm{Re}\{\hat{k}_\mathrm{C}(\boldsymbol{x}, \boldsymbol{y})\} = \hat{k}_\mathrm{CtR}(\boldsymbol{x}, \boldsymbol{y}). \tag{2.23}$$

We call Φ_CtR a *Complex-to-Real (CtR)* polynomial random feature and summarize its construction in Algorithm (1).
We provide the following theorems showing the variance reduction for Rademacher and Gaussian sampling and that are proven in [Wacker, 2022b]. In the following, $\mathbb{V}[\cdot]$ denotes the variance.

Algorithm 1 Complex-to-Real (CtR) Random Features

Input : Datapoint $\boldsymbol{x} \in \mathbb{R}^d$
Choose projection dimension $D \in \mathbb{N}$, degree $p \in \mathbb{N}$
Sample $\{\boldsymbol{W}_i\}_{i=1}^p$ with $\boldsymbol{W}_i \in \mathbb{C}^{D\times d}$ independently according to one of these random features :

- Gaussian : $(\boldsymbol{W}_i)_{\ell,k} \overset{i.i.d.}{\sim} \mathcal{CN}(0,1)$
- Rademacher : $(\boldsymbol{W}_i)_{\ell,k} \overset{i.i.d.}{\sim} \mathrm{Unif}(\{1,-1,\mathrm{i},-\mathrm{i}\})$

Compute $\Phi_{\mathrm{C}}(\boldsymbol{x}) := (\boldsymbol{W}_1\boldsymbol{x} \odot \cdots \odot \boldsymbol{W}_p\boldsymbol{x})/\sqrt{D}$
Return :
$\Phi_{\mathrm{CtR}}(\boldsymbol{x}) := (\mathrm{Re}\{\Phi_{\mathrm{C}}(\boldsymbol{x})_1\},\dots,\mathrm{Re}\{\Phi_{\mathrm{C}}(\boldsymbol{x})_D\},\mathrm{Im}\{\Phi_{\mathrm{C}}(\boldsymbol{x})_1\},\dots,\mathrm{Im}\{\Phi_{\mathrm{C}}(\boldsymbol{x})_D\})^\top \in \mathbb{R}^{2D}$

Theorem 2.2.7 (CtR-Rademacher advantage)**.** *Let* $a := \sum_{i=1}^d \sum_{j\neq i}^d x_i x_j y_i y_j$ *and* $b(j) := \|\boldsymbol{x}\|^{2j}\|\boldsymbol{y}\|^{2j} - (\sum_{i=1}^d x_i^2 y_i^2)^j \geq 0$. *Then* $\mathbb{V}[\hat{k}_{\mathrm{R}}(\boldsymbol{x},\boldsymbol{y})] - \mathbb{V}[\hat{k}_{\mathrm{CtR}}(\boldsymbol{x},\boldsymbol{y})]$ *yields*

$$\frac{1}{2D}\sum_{k=2}^{p}\sum_{j=0}^{k-1}\binom{p}{k}\binom{k}{j} b(j)\ a^{p-j} \geq 0 \quad \textit{if} \quad a \geq 0.$$

Furthermore, CtR-Rademacher sketches achieve the lowest possible variance for $\hat{k}_{\mathrm{CtR}}(\boldsymbol{x},\boldsymbol{y}) = \mathrm{Re}\{\Phi_{\mathrm{C}}(\boldsymbol{x})^\top \overline{\Phi_{\mathrm{C}}(\boldsymbol{y})}\}$ *with* Φ_{C} *being defined through Equation 2.21.*

Theorem 2.2.8 (CtR-Gaussian advantage)**.** *For any* $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^d$, $\mathbb{V}[\hat{k}_{\mathrm{R}}(\boldsymbol{x},\boldsymbol{y})] - \mathbb{V}[\hat{k}_{\mathrm{CtR}}(\boldsymbol{x},\boldsymbol{y})]$ *yields*

$$\frac{1}{2D}\sum_{k=0}^{p-1}\binom{p}{k}(2^k-1)(\boldsymbol{x}^\top\boldsymbol{y})^{2k}\left(\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2\right)^{p-k} \geq 0.$$

Thus, regardless of the input data, Φ_{CtR} should be preferred over Φ_{R} when using Gaussian polynomial random features. The advantage again increases with p.
In [Wacker, 2022b], the CtR feature map was also combined with Orthogonal random features and the TensorSRHT algorithm in order to lower its computational complexity.

2.3 Deep neural networks and training methods

Deep neural networks. During the process of learning a function minimizing the empirical risk, we could simply parameterize the function by a neural network. This function will be the combination of composed functions and matrix multiplication.
Forward pass. In a model with L layers of neurons, $\ell \in \{1,\dots,L\}$ is the index of the ℓ-th layer, $\mathbf{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ the weight matrix between layers $\ell-1$ and ℓ, ϕ_ℓ the activation function of the neurons, and $\mathbf{h}_\ell$ their activations. The forward pass for a pair $(\boldsymbol{x}, y)$ writes as :

$$\forall \ell \in \{1,\dots,L\} : \mathbf{z}_\ell = \mathbf{W}_\ell \mathbf{h}_{\ell-1}, \mathbf{h}_\ell = \phi_\ell(\mathbf{z}_\ell), \tag{2.24}$$

where $\mathbf{h}_0 \doteq \boldsymbol{x}$ is the data and $\hat{\boldsymbol{y}} \doteq \mathbf{h}_L = \phi_L(\mathbf{z}_L)$ is the predicted output. The network is trained by defining a loss function $\mathcal{L}$, measuring the discrepancy between the true label $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$.

Before a feedforward network, inductive bias can be incorporated such as convolutional layers [Lecun, 1998](made for extracting patterns in images) or attention layers [Bahdanau, 2014 ; Vaswani, 2017](which can extract any correlation between dimensions of the datapoints, and they can be seen as a generalization of convolutions).

Training methods. We will now develop how to optimize the neural network. The standard method is backpropagation (BP), but as will we show, new more biologically realist methods such as Direct Feedback Alignment (DFA) have been developed.

Proposition 2.3.1 (Chain rule)**.** *If a variable z depends on a variable y which itself depends on a variable x, then the derivative of z with respect to x can be expressed thanks to the chain rule :*

$$\frac{dz}{dx} = \frac{dz}{dy}.\frac{dy}{dx}. \tag{2.25}$$

Backpropagation. To update a weight matrix $\mathbf{W}_\ell$ for minimizing the loss $\mathcal{L}$ at the end of the network, we should go into the direction of the loss minimized by this weight. The stepsize is called the *learning rate* η. With backpropagation [Rumelhart, 1986], the weight updates are computed using the chain rule of derivatives (see Proposition 2.3.1), yielding a change $\delta\mathbf{W}_\ell$ of the weights of layer ℓ between two iterations :

$$\delta\mathbf{W}_\ell = -\eta\frac{\partial\mathcal{L}}{\partial\mathbf{W}_\ell} = -\eta\frac{\partial\mathcal{L}}{\partial\mathbf{h}_\ell}\frac{\partial\mathbf{h}_\ell}{\partial\mathbf{z}_\ell}\frac{\partial\mathbf{z}_\ell}{\partial\mathbf{W}_\ell} \tag{2.26}$$

$$= -\eta[((\mathbf{W}_{\ell+1})^\top\delta\mathbf{z}_{\ell+1}) \odot \phi'_\ell(\mathbf{z}_\ell)](\mathbf{h}_{\ell-1})^\top, \tag{2.27}$$

where $\delta\mathbf{z}_\ell = \frac{\partial\mathcal{L}}{\partial\mathbf{z}_\ell}$, ϕ'_ℓ is the derivative of ϕ_ℓ, $\odot$ is the Hadamard product, and $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ is the prediction loss.

2.3.1 (Direct) Feedback Alignment

DFA is a biologically inspired alternative to backpropagation with an asymmetric backward pass. For ease of notation, we introduce it for fully connected networks but it also generalizes to transformers and other architectures [Launay, 2020a]. It has been theoretically studied in [Lillicrap, 2016 ; Refinetti, 2021].

Feedback Alignment replaces the weight matrix $(\mathbf{W}_{\ell+1})^\top$ by a random matrix $\mathbf{B}_\ell$ of the same dimensions. This is supposed to be more biologically plausible, as supported in [Lillicrap, 2014 ; Lillicrap, 2016] because neurons in the brain are not supposed to memorize the information that go forward and transmit it backward ; they rather send random signals to learn.

$$\delta\mathbf{W}_\ell^{\text{FA}} = -\eta[(\mathbf{B}_\ell\delta\mathbf{z}_{\ell+1}) \odot \phi'_\ell(\mathbf{z}_\ell)](\mathbf{h}_{\ell-1})^\top, \delta\mathbf{z}_{\ell+1} = \frac{\partial\mathcal{L}}{\partial\mathbf{z}_{\ell+1}}. \tag{2.28}$$

Direct Feedback Alignment replaces the gradient signal $(\mathbf{W}_{\ell+1})^\top\delta\mathbf{z}_{\ell+1}$ with a random projection of the derivative of the loss with respect to the pre-activations $\delta\mathbf{z}_L$ of the last layer. For losses $\mathcal{L}$ commonly used in classification and regression, such as the squared loss or the

cross-entropy loss, this will amount to a random projection of the error $\mathbf{e} \propto \hat{\boldsymbol{y}} - \boldsymbol{y}$. With a fixed random matrix $\mathbf{B}_\ell$ of appropriate shape drawn at initialization of the learning process, the parameter update of DFA is :

$$\delta \mathbf{W}_\ell^{\text{DFA}} = -\eta[(\mathbf{B}_\ell \mathbf{e}) \odot \phi'_\ell(\mathbf{z}_\ell)](\mathbf{h}_{\ell-1})^\top, \mathbf{e} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}_L}. \tag{2.29}$$

Backpropagation vs DFA training. Learning using backpropagation consists in iteratively applying the forward pass Equation 2.24 on batches of training examples and then applying backpropagation updates Equation 2.27. Training with DFA consists in replacing the backpropagation updates by DFA ones Equation 2.29. An interesting feature of DFA is the parallelization of the training step, where all the random projections of the error can be done at the same time, as illustrated in Figure 2.1. Indeed, updating the weights of layer ℓ does not depend on the weights of layer $\ell + 1$ to be updated.

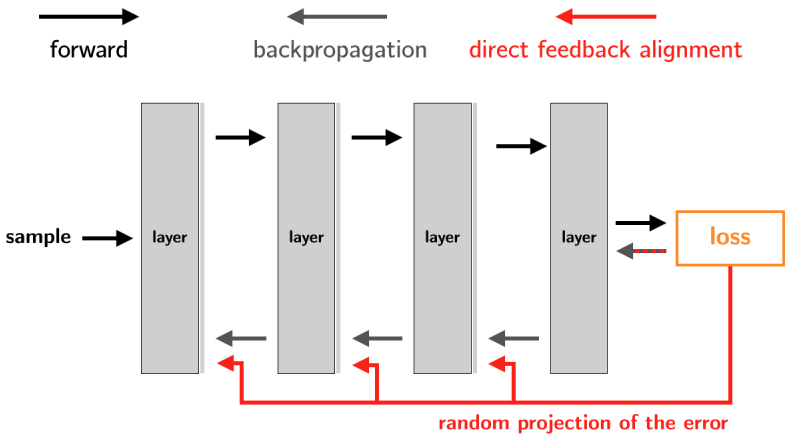


FIGURE 2.1 – Schematic comparison of backpropagation and Direct Feedback Alignment. The two approaches differ in how the loss impacts each layer of the model. While in backpropagation, the loss is propagated sequentially backwards, in DFA, it directly acts on each layer after random projection.

2.3.1.1 Minor contribution : align then memorize, the dynamics of learning with (Direct) Feedback Alignment

This section is an extract of [Refinetti, 2021].

Despite relying on random feedback weights for the backward pass, DFA successfully trains state-of-the-art models such as Transformers [Launay, 2020a]. On the other hand, it notoriously fails to train convolutional networks. An understanding of the inner workings of DFA to explain these diverging results remains elusive. Here, we propose a theory of feedback alignment algorithms. We first show that learning in shallow networks proceeds in two steps : an *alignment* phase, where the model adapts its weights to align the approximate gradient with the true gradient of the loss function, is followed by a *memorisation* phase, where the model focuses on fitting the

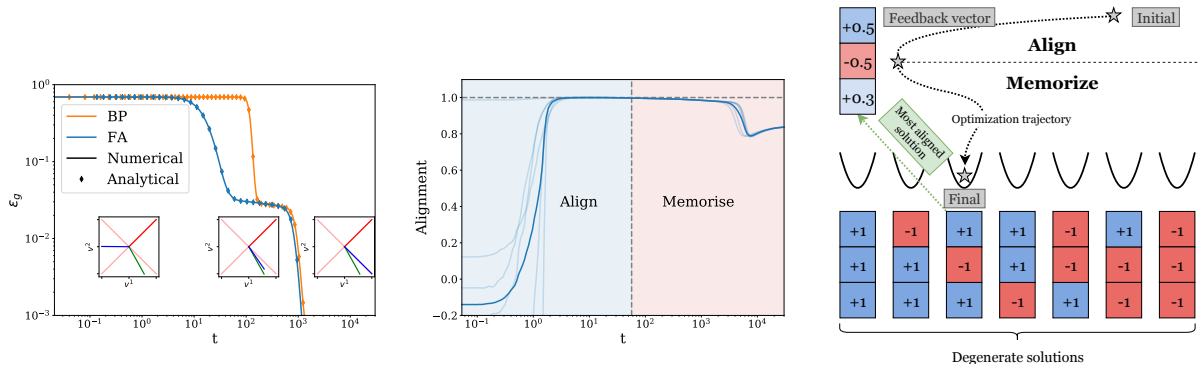
data. This two-step process has a *degeneracy breaking* effect : out of all the low-loss solutions in the landscape, a network trained with DFA naturally converges to the solution which maximises gradient alignment.

Related Work. [Lillicrap, 2016] gave a first theoretical characterisation of feedback alignment by arguing that for two-layer linear networks, FA works because the transpose of the second layer of weights $\mathbf{W}_2$ tends to align with the random feedback matrix $\mathbf{B}_1$ during training. This *weight alignment* (WA) leads the weight updates of FA to align with those of BP, leading to *gradient alignment* (GA) and thus to successful learning. [Frenkel, 2019] extended this analysis to the deep linear case for a variant of DFA called "Direct Random Target Projection" (DRTP), under the restrictive assumption of training on a single data point. [Nøkland, 2016a] also introduced a layerwise alignment criterion to describe DFA in the deep nonlinear setup, under the assumption of constant update directions for each data point.
We begin with an exact description of DFA dynamics in shallow non-linear networks. Here we consider a high-dimensional scalar regression task where the inputs $\boldsymbol{x} \in \mathbb{R}^d$ are sampled i.i.d. from the standard normal distribution. We focus on the classic *teacher-student* setup, where the labels $y \in \mathbb{R}$ are given by the outputs of a "teacher" network with random weights [Gardner, 1989 ; Seung, 1992 ; Watkin, 1993 ; Engel, 2001 ; Zdeborová, 2016]. In this section, we let the input dimension $d \to \infty$, while both teacher and student are two-layer networks with $K, M \sim O(1)$ hidden nodes.
We consider sigmoidal, $g(x) = \text{erf}\,(x/\sqrt{2})$, and ReLU activation functions, $g(x) = \max(0, x)$. We asses the student's performance on the task through its the *generalisation error*, or test error :

$$\varepsilon_g(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \equiv \frac{1}{2}\mathbb{E}\left[\hat{y} - y\right]^2 \equiv \frac{1}{2}\mathbb{E}[e^2], \tag{2.30}$$

where the expectation $\mathbb{E}$ is taken over the inputs for a given teacher and student networks with parameters $\tilde{\boldsymbol{\theta}} = (M, \tilde{\mathbf{W}}_1, \tilde{\mathbf{W}}_2, g)$ and $\theta = (K, \mathbf{W}_1, \mathbf{W}_2, g)$. Learning a target function such as the teacher is a widely studied setup in the theory of neural networks [Zhong, 2017 ; Advani, 2020 ; Tian, 2017 ; Du, 2018 ; Soltanolkotabi, 2018 ; Aubin, 2018 ; Saxe, 2018 ; Baity-Jesi, 2018 ; Goldt, 2019 ; Ghorbani, 2019 ; Yoshida, 2019 ; Bahri, 2020 ; Gabrié, 2020].

An analytical theory for DFA dynamics To better understand DFA, we study its dynamics in the limit of infinite training data where a previously unseen sample $(\boldsymbol{x}, y)$ is used to compute the DFA weight updates of Equation 2.29 at every step. This "online learning" or "one-shot/single-pass" limit of SGD has been widely studied in recent and classical works on vanilla BP [Kinzel, 1990 ; Biehl, 1995 ; Saad, 1995a ; Saad, 1995b ; Saad, 2009 ; Zhong, 2017 ; Brutzkus, 2017 ; Mei, 2018 ; Rotskoff, 2018 ; Chizat, 2018 ; Sirignano, 2019].
We work in the regime where the input dimension $d \to \infty$, while M and K are finite. The test error Equation 2.30, i.e. a function of the student and teacher parameters involving a high-dimensional average over inputs, can be simply expressed in terms of a *finite* number of



(a) Generalization dynamics (b) Alignment dynamics (c) Degeneracy breaking

FIGURE 2.2 – *(a)* **Theory gives exact prediction for the learning dynamics.** We plot learning curves for BP and DFA obtained from (i) a single simulation (solid lines), (ii) integration of the ODEs for BP dynamics [Biehl, 1995 ; Saad, 1995a] (orange dots), (iii) integration of the ODEs for DFA derived here (blue dots). *Insets* : Teacher second-layer weights (red) as well as the degenerate solutions (light red) together with the feedback vector F_1 (green) and the student second-layer weights v (blue) at three different times during training with DFA. *Parameters :* $d = 500, K = M = 2, \eta = 0.1, \sigma_0 = 10^{-2}$ (the standard deviation of initialized weights).
(b) **Align-then-Memorise process.** Alignment (cosine similarity) between the student's second layer weights and the feedback vector. In the align phase, the alignment increases, and reaches its maximal value when the test loss reaches the plateau. Then it decreases in the memorization phase, as the student recovers the teacher weights.
(c) **The degeneracy breaking mechanism.** There are multiple degenerate global minima in the optimisation landscape : they are related through a discrete symmetry transformation of the weights that leaves the student's output unchanged. DFA chooses the solution which maximises the alignment with the feedback vector.

"order parameters" $Q = (Q^{kl}), R = (R^{km}), T = (T^{mn})$,

$$\lim_{d\to\infty} \varepsilon_g(\theta, \tilde{\theta}) = \varepsilon_g(Q, R, T, \mathbf{W}_2, \tilde{\mathbf{W}}_2), \tag{2.31}$$

where

$$Q^{kl} = \frac{W_1^k W_1^l}{d}, \quad R^{km} = \frac{W_1^k \tilde{W}_1^m}{d}, \quad T^{mn} = \frac{\tilde{W}_1^m \tilde{W}_1^n}{d}, \tag{2.32}$$

as well as second layer weights $\tilde{W}_2^m$ and W_2^k [Saad, 1995a ; Saad, 1995b ; Biehl, 1995 ; Engel, 2001]. Intuitively, R^{km} quantifies the similarity between the weights of the student's kth hidden unit and the teacher's mth hidden unit. The self-overlap of the kth and lth student nodes is given by Q^{kl}, and likewise T^{mn} gives the (static) self-overlap of teacher nodes. In seminal work, [Saad, 1995a] and [Biehl, 1995] obtained a closed set of ordinary differential equations (ODEs) for the time evolution of the order parameters Q and R. Our first main contribution is to extend their approach to the DFA setup (not developed here, see [Refinetti, 2021] for the exact computations), obtaining a set of ODEs that predicts the test error of a student trained using DFA (see Equation 2.29) at all times. The accuracy of the predictions from the ODEs is demonstrated in Figure 2.2 (a), where the comparison between a single simulation of training a two-layer net with BP (orange) and DFA (blue) and theoretical predictions yield perfect agreement.

Sigmoidal networks learn through "degeneracy breaking". The test loss of a sigmoidal student trained on a teacher with the same number of neurons as herself ($K = M$) contains several global minima, which all correspond to fixed points of the ODEs (see Section A.7 of [Refinetti, 2021]). Among these is a student with exactly the same weights as her teacher. The symmetry $\text{erf}(z) = -\text{erf}(-z)$ induces a student with weights $\{\tilde{\mathbf{W}}_1, \tilde{\mathbf{W}}_2\}$ to have the same test error as a sigmoidal student with weights $\{-\tilde{\mathbf{W}}_1, -\tilde{\mathbf{W}}_2\}$. Thus, as illustrated in Figure 2.2 (c), the problem of learning a teacher has various degenerate solutions. A student trained with vanilla BP converges to any one of these solutions, depending on the initial conditions.

Alignment phase A student trained using DFA has to fulfil the same objective (zero test error), with an additional constraint : her second-layer weights $\mathbf{W}_2$ need to align with the feedback vector $\mathbf{B}_1$ to ensure the first-layer weights are updated in the direction that minimises the test error. And indeed, an analysis of the ODEs (see Section B of [Refinetti, 2021]) reveals that in the early phase of training, $\dot{\mathbf{W}}_2 \sim \mathbf{B}_1$ and so $\mathbf{W}_2$ grows in the direction of the feedback vector $\mathbf{B}_2$ resulting in an increasing overlap between $\mathbf{W}_2$ and $\mathbf{B}_1$. In this *alignment phase* of learning, shown in Figure 2.2 (b), $\mathbf{W}_2$ becomes perfectly aligned with $\mathbf{B}_1$. DFA has perfectly recovered the weight updates for $\mathbf{W}_1$ of BP, but the second layer has lost its expressivity (it is simply aligned to the random feedback vector).

Memorisation phase The expressivity of the student is restored in the *memorisation* phase of learning, where the second layer weights move away from $\mathbf{B}_1$ and towards the global minimum of the test error that maintains the highest overlap with the feedback vector. In other words, students solve this constrained optimisation problem by consistently converging to the global minimum of the test loss that simultaneously maximises the overlap between $\mathbf{W}_2$ and $\mathbf{B}_1$, and thus between the DFA gradient and the BP gradient. For DFA, the global minima of the test loss are not equivalent, this "degeneracy breaking" is illustrated in Figure 2.2 (c).
In [Refinetti, 2021], we also identify a key quantity underlying alignment in deep linear networks : the conditioning of the *alignment matrices*. The latter enables a detailed understanding of the impact of data structure on alignment, and suggests a simple explanation for the well-known failure of DFA to train convolutional neural networks. Numerical experiments on MNIST and CIFAR-10 clearly demonstrate degeneracy breaking in deep non-linear networks and show the align-then-memorize process occurs sequentially from the bottom layers of the network to the top.

2.4 Optical Processing Units

2.4.1 Optical random features

Increasing the number of random features D for kernel approximation can quickly become expensive since there the complexity of constructing a random feature is $O(ndD)$. This issue led to the development of a new optical hardware, leveraging million-scale projection.
The principle of the random projections performed by the *Optical Processing Unit* (OPU) is based on the use of a heterogeneous material that scatters the light going through it, see Figure 2.3 for the experimental setup. The data vector $\boldsymbol{x} \in \mathbb{R}^d$ is encoded into light using a

digital micromirror device (DMD), which is an ensemble of millions of micro-mirrors which can be turned on and off. This encoded light then passes through the heterogeneous medium, performing the random matrix multiplication. As demonstrated in [Liutkus, 2014], light going through the scattering medium follows many extremely complex paths, that depend on refractive index inhomogeneities at random positions. For a fixed scattering medium, the resulting process is still linear, deterministic, and reproducible. Reproducibility is important as all our data vectors need to be multiplied by the same realisation of the random matrix.

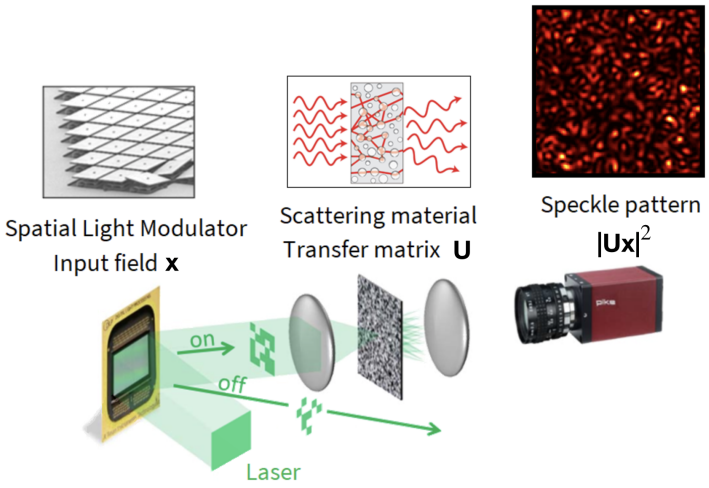


FIGURE 2.3 – Experimental setup of the Optical Processing Unit. The data vector is encoded in the coherent light from a laser using a DMD. Light then goes through a scattering medium and a speckle pattern is measured by a camera.

After going through the "random" medium, we observe a speckle figure on the camera, where the light intensity at each point is modelled by a sum of the components of $\boldsymbol{x}$ weighted by random coefficients. See [Brossollet, 2021 ; Poli, 2021] for more details. Measuring the intensity of light induces a non-linear transformation of this sum, leading to :

Proposition 2.4.1. *Given a data vector $\boldsymbol{x} \in \mathbb{R}^d$, the random feature map performed by the Optical Processing Unit is :*

$$\boldsymbol{\phi}(\boldsymbol{x}) = \frac{1}{\sqrt{D}}|\mathbf{U}\boldsymbol{x}|^2 \quad \in \mathbb{R}^D, \tag{2.33}$$

where $\mathbf{U} \in \mathbb{C}^{D\times d}$ is a complex Gaussian random matrix whose elements $U_{i,j} \sim \mathcal{CN}(0,1)$, the variance being set to one without loss of generality, and depends on a multiplicative factor combining laser power and attenuation of the optical system. We will name these RFs optical random features.

As this feature map is generated optically and then treated numerically on the computer, the exponent of the feature map can be changed and any mathematical operation can be performed at the top of Equation 2.33.

Linear transform. In the flavour of optical holography [Yamaguchi, 2006], a digital holographic operation can be performed, allowing to get rid of the non-linearity. This technique was developed in the framework of this thesis, but is confidential and briefly discussed in the patent [Poli, 2021].

This yields the following operation :

$$\phi(\boldsymbol{x}) = \frac{1}{\sqrt{D}}\mathbf{U}'\boldsymbol{x}, \tag{2.34}$$

where $\mathbf{U}' \in \mathbb{R}^{D\times d}$ is a real Gaussian random matrix whose elements $\mathbf{U}'_{i,j} \sim \mathcal{N}(0,1)$, the variance being set to one without loss of generality.
As seen on Figure 2.3, the input light bounces on an ensemble of micro-mirrors, which are either on or off, encoding a 0 or a 1. It means that in practice, the data has to be encoded in binary, possible thanks to the following encoding methods :

- **Thresholding :** we can set a threshold value t where if $\boldsymbol{x}_i > t$ with $i = 1, ...d$, it is set to 1, otherwise, set to 0.
- **Learned threshold :** the threshold value can be seen as a parameter to learn via cross-validation
- **Training an auto-encoder :** An auto-encoder (AE) can be trained where the binarization is in the latent space. We would just have to train the AE and extract the encoder part to binarize our data. The binarization would therefore be data-dependent.
- **Bit-plan encoding :** in the case of the linear transform of Equation 2.34, we can chose a number of bits for encoding our data. Because images are usually encoded in 8 bits, we can keep the full information of the image easily.
- **Using reinforce algorithm :** one can incorporate the OPU inside an AE and bypass it using the REINFORCE algorithm [Kozyrskiy, 2021] to learn a binarization for it.

2.5 Security of systems

Open source packages facilitated the wide adoption of machine learning algorithms, and many companies adopted a transformation of their prediction systems. However, these systems can be the target of attackers, who may either fool the systems or retrieve information about the data. Indeed, machine learning systems can even be the *weakest link* to attack in the security chain whose vulnerability can compromise the whole infrastructure.
These weaknesses have led to new research topics in machine learning, either in defending the algorithm against fooling attacks (adversarial robustness) or protecting the information about users in the dataset, i.e. yielding privacy (differential privacy). In the following, we will develop some basics concepts.

2.5.1 Adversarial robustness

The philosophy of adversarial attacks is to find a data (and architecture) dependent perturbation $\boldsymbol{\delta}$, and that will fool the neural network (represented by the function f_θ) to attack. It aims at maximizing the loss ℓ with a perturbation added to the data, without being visible by the human eye, implying to restrict the magnitude of this perturbation. In practice, this is performed using

gradient ascent on the perturbation, or more formally, it aims at solving :

$$\boldsymbol{\delta} = \underset{\|\boldsymbol{\delta}\| \leq \varepsilon}{\arg\max}\ \ell(f_\theta(\boldsymbol{x} + \boldsymbol{\delta}), y). \tag{2.35}$$

White-box attacks suppose the attacker has a full knowledge of the model and its parameters, and can therefore generate adversarial example using gradient-based methods.
Fast-gradient sign method (FGSM) [Goodfellow, 2015] is the simplest method to attack a network with parameters θ. It simply consists in computing the gradient of the loss function and adding to the attacked image its sign times a perturbation ε. The adversarially perturbed example $\tilde{\boldsymbol{x}}$ of an input $\boldsymbol{x}$ associated to a label y is computed as :

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\delta} = \boldsymbol{x} + \varepsilon.\text{sign}(\nabla_{\boldsymbol{x}} \ell(f_\theta(\boldsymbol{x}), y)). \tag{2.36}$$

Projected Gradient Descent (PGD) [Madry, 2018a] is an extension of FGSM where the adversarial example is iteratively optimized (iterations denoted by t) with the following equation :

$$\boldsymbol{x}^{t+1} = \Pi_{B_\infty(\boldsymbol{x},\varepsilon)} \left[\boldsymbol{x}^t + \alpha.\text{sign}\left(\nabla_{\boldsymbol{x}} \ell(f_\theta(\boldsymbol{x}^t), y) \right) \right], \tag{2.37}$$

where $\Pi_{B_\infty(\boldsymbol{x},\varepsilon)}$ is the orthogonal projection on $B_\infty(\boldsymbol{x}, \varepsilon) := \{\boldsymbol{x}' : \|\boldsymbol{x}' - \boldsymbol{x}\|_\infty \leq \varepsilon\}$ and $\boldsymbol{x}^0 = \boldsymbol{x}$. The quantity ε is the strength of the perturbation and α the equivalent of a learning rate.
Black-box attacks are more realistic settings, where the attacker does not have access to information about the neural networks to attack. For instance, if the attacker wants to attack a road sign recognition system of a self-driving car, the algorithm is unknown, but good adversarial examples can still fool the car [Ren, 2019 ; Kumar, 2020]. This could be critical in the real-world. In practice, the goal of black-box attacks is to perform estimation of the gradient of the network with respect to queries.
Natural evolution strategy (NES) [Wierstra, 2014 ; Ilyas, 2018a] are black-box attacks crafted for query limited attacks. It consists in efficiently estimating the gradient of the loss with respect to the input $\boldsymbol{x}$:

$$\begin{aligned} \boldsymbol{\delta} = \nabla_{\boldsymbol{x}} \ell(\boldsymbol{x}, y) &\simeq \nabla_{\boldsymbol{x}} \mathbb{E}_{\mathbf{u} \sim \mathcal{N}(\boldsymbol{x}, \sigma^2 \mathbf{I})} \left[\ell(\mathbf{u}, y) \right] \\ &\simeq \frac{1}{\sigma N} \sum_{i=1}^{N} \mathbf{g_i} \ell(\boldsymbol{x} + \sigma \mathbf{g_i}, y), \end{aligned}$$

where y is the true label, $\mathbf{g_i}$ are standard Gaussian random vectors and N the sample size of the Monte-Carlo estimation.
Bandit attacks [Ilyas, 2018b] are an extension of NES attacks, taking advantage of gradients correlations for close pixels and between gradient steps.
Parsimonious attacks [Moon, 2019a] propose an efficient discrete surrogate to the optimization problem for attacking the network which does not require estimating the gradient. The attacks becomes free of first order update of the hyperparameters to tune and is stated as a combinatorial problem to solve. These attacks are known to be more efficient than NES and Bandits attacks.

Transfer attacks. When the attacker have access to limited number of queries to the system to attack and doesn't know its parameters, a transfer attack can be used to fool it. The latter consists in building a network onto which the attackers creates optimal perturbations using a white-box attack such as PGD. Then, these perturbations are transferred to the network to attack in order to fool it.

Adversarial training [Madry, 2018a] is an adversarial defense, i.e. a way to make our algorithm robust to adversarial attacks. The principle is to first learn a perturbation on our network using FGSM or PGD attacks, and then train our network on an attacked training set. Formally, it consists in solving on a training set $\mathcal{S}$:

$$\min_{\theta} \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{x},y \in \mathcal{S}} \max_{\|\boldsymbol{\delta}\| \leq \varepsilon} \ell(f_\theta(\boldsymbol{x} + \boldsymbol{\delta}), y). \tag{2.38}$$

This is efficient for making our network robust against standard attacks. However, there is a trade-off with the accuracy. The more robust the network, the less accurate are its predictions. This is why a lot of new adversarial methods have emerged, and have been gathered on Robustbench [1] [Croce, 2020a], aiming at standardizing the evaluation of a defense on an ensemble of attacks named AutoAttack [Croce, 2020c].

2.5.2 Differential privacy

Differential privacy (DP) is a framework used to protect the privacy of individuals gathered in a dataset. It is not an algorithm in itself, but rather a property that multiple methods can achieve, and this property mathematical quantifies the privacy risk.

By setting a *privacy loss parameter*, i.e. a maximum level of privacy loss allowed, DP aims at manipulating the dataset or the algorithm processing it in order to achieve this level of privacy, while maintaining the utility/accuracy on the dataset. DP is robust against a variety of privacy attacks, but is also transparent, meaning it has the ability to share statistics about the dataset without affecting every individual's privacy. This opens the path for the sharing of sensitive data that could not be shared previously, such as medical or financial data.

However, these benefits come with several challenges. The decrease in accuracy can be larger for small datasets than for larger ones, and this decrease is hard to bypass since there is the existence of a privacy/accuracy trade-off to take into account. Nevertheless, it started to grow in popularity, from the private (Facebook, Amazon, Uber, Apple) to the public sector (U.S. Census Bureau) where it was used to protect sensitive data against potential privacy attacks.

In practice, Differential Privacy is created by introducing noise – that can be perceived as random information – in the system (at the data level, or the algorithm one) so it becomes increasingly difficult to tell if a specific individual's information was used or not. This lead to a less accurate prediction, that is approximately the same as the true value.

Informal Example. To better understand the logic behind Differential Privacy, let us give an informal example. Let's suppose you want to share the average salary among 100 employees.

1. https ://robustbench.github.io/

You can query to the system and obtain the average salary without revealing information about individual employees. Now, let's say you get rid of the information of an individual and get the average salary about the 99 others. Then, by simple algebra, you could deduct the missing individual's salary.
Now, let's imagine the information is shared using a differentially private query system. When a malicious actors queries the database, the algorithm return the truth (i.e. the average salary) plus some random noise (different realization of the noise at each query). Now, if the malicious actor knew everything about the 99 other employees, he/she can't be sure about the missing individual's salary, whose privacy is protected. In this case, it allowed us to release statistics about a dataset without sharing private information about single individuals.
Mathematical framework. Let's develop more formally the mathematical framework of Differential Privacy by introducing some useful definitions :

Definition 2.5.1 (Neighboring datasets)**.** *Let* $\{\mathcal{X}^j\}_{j=1}^N$ *(e.g.* $\mathcal{X}^j = \mathbb{R}^d$*) be a* domain *and* $\mathcal{D} \doteq \cup_{j=1}^N \mathcal{X}^j$*.* $D, D' \in \mathcal{D}$ *are* neighboring datasets *if they differ from one element. This is denoted by* $\mathcal{D} \sim \mathcal{D}'$*.*

Definition 2.5.2 (ε-DP [Dwork, 2006b])**.** *A randomized algorithm* $\mathcal{A} : \mathcal{D} \to Range(\mathcal{A})$ *satisfies* ε*-differential privacy (*ε*-DP) if for any neighboring datasets* $D, D' \in \mathcal{D}$ *and* $\mathcal{O} \subseteq Im(\mathcal{A})$*,*

$$\mathbb{P}[\mathcal{A}(D) \in \mathcal{O}] \leq e^{\varepsilon}\mathbb{P}[\mathcal{A}(D') \in \mathcal{O}]. \tag{2.39}$$

A mechanism that satisfies the ε-DP definition is the addition of Laplace noise, called the *Laplace mechanism*, which allows the release of a noisy answer to an arbitrary query to the algorithm computing a function f, and is defined as followed :

$$\boldsymbol{L}_{\varepsilon,f} \doteq f(\boldsymbol{x}) + \Lambda\left(0, \frac{\Delta_{1,f}}{\varepsilon}\right), \tag{2.40}$$

where Λ is the Laplace distribution and $\Delta_{p,f}$ is the ℓ_p-sensitivity of the algorithm f defined as :

Definition 2.5.3 (ℓ_p-sensitivity)**.** *Let* D *and* D' *be neighboring datasets. Then the* ℓ_p*-sensitivity of a function* f *is defined as*

$$\Delta_{p,f} \doteq \max_{D,D'} \|f(D) - f(D')\|_p \,. \tag{2.41}$$

If f and g are ε_1 and ε_2-DP respectively, then the basic composition theorem [Mironov, 2017] states that the simultaneous release of $f(D)$ and $g(D)$ is $(\varepsilon_1 + \varepsilon_2)$-DP. A relaxation of ε-DP allows a δ additive term in the defining inequality :

Definition 2.5.4 ((ε, δ)-differential privacy [Dwork, 2008])**.** *Let* $\varepsilon, \delta > 0$*. Let* $\mathcal{A} : \mathcal{D} \to Range(\mathcal{A})$ *be a* randomized *algorithm, where* $Range(\mathcal{A})$ *is the range of* $\mathcal{D}$ *through* $\mathcal{A}$*.* $\mathcal{A}$ *is* (ε, δ)*-differentially private, or* (ε, δ)*-DP, if for all neighboring datasets* $D, D' \in \mathcal{D}$ *and for all sets* $\mathcal{O} \in Im\ \mathcal{A}$*, the following inequality holds :*

$$\mathbb{P}[\mathcal{A}(D) \in \mathcal{O}] \leq e^{\varepsilon}\,\mathbb{P}[\mathcal{A}(D') \in \mathcal{O}] + \delta,$$

where the probability relates to the randomness of $\mathcal{A}$.

An interpretation is that it is ε-DP, excepted with probability δ. However, as it is explained in [Mironov, 2017], it is *qualitatively* different than pure ε-DP (unless of course $\delta = 0$).
The definition of (ε, δ)-DP was initially proposed to capture privacy guarantees to the Gaussian mechanism :

Definition 2.5.5 (Gaussian mechanism [Mironov, 2019])**.** *Let* D *and* D' *be neighboring dataset. Then the Gaussian mechanism is defined as*

$$\mathcal{M}_\sigma f(\boldsymbol{x}) = f(\boldsymbol{x}) + \mathcal{N}(0, \sigma^2 \mathbf{I}). \tag{2.42}$$

An elementary analysis shows that the Gaussian mechanism does not meet ε-DP for any ε. Instead, it satisfies a continuum of incomparable (ε, δ)-DP guarantees, for all $\varepsilon < 1$ and $\sigma > \sqrt{2 \ln \frac{1.25}{\delta}} \frac{\Delta_{2,f}}{\varepsilon}$.
[Mironov, 2017] proposed an alternative notion of differential privacy based on Rényi α-divergences and established a connection between their definition and the (ε, δ)-differential privacy of Definition 2.5.4. Rényi-based Differential Privacy is captured by the following :

Definition 2.5.6 (Rényi α-divergence [Rényi, 1961])**.** *For two probability distributions* P *and* Q *defined over* $\mathbb{R}$*, the Rényi divergence of order* $\alpha > 1$ *is given by :*

$$\mathbb{D}_\alpha \left(P \| Q\right) \doteq \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim Q} \left(\frac{P(x)}{Q(x)} \right)^\alpha . \tag{2.43}$$

Definition 2.5.7 ((α, ε)-Rényi differential privacy [Mironov, 2017])**.** *Let* $\varepsilon > 0$ *and* $\alpha > 1$*. A randomized algorithm* $\mathcal{A}$ *is* (α, ε)*-Rényi differential private or* (α, ε)*-RDP, if for any neighboring datasets* $D, D' \in \mathcal{D}$*,*

$$\mathbb{D}_\alpha \left(\mathcal{A}(D) \| \mathcal{A}(D')\right) \leq \varepsilon.$$

Theorem 2.5.8 (Composition of RDP mechanisms [Mironov, 2017])**.** *Let* $\{M_i\}_{i=1}^k$ *be a set of mechanisms, each satisfying* (α, ε_i)*-RDP. Then their combination is* $(\alpha, \sum_i \varepsilon_i)$*-RDP.*

Going from RDP to the Differential Privacy of Definition 2.5.4 is made possible by the following theorem (see also [Asoodeh, 2020 ; Balle, 2018 ; Wang, 2019]) :

Theorem 2.5.9 (Converting RDP parameters to DP parameters [Mironov, 2017])**.** *An* (α, ε)*-RDP mechanism is* $\left(\varepsilon + \frac{\log 1/\delta}{\alpha - 1}, \delta\right)$*-DP for all* $\delta \in (0, 1)$.

2.6 PAC-Bayes Bounds

This section is based on the excellent introduction to PAC-Bayes bounds of [Alquier, 2021]. Let us place ourselves in the supervised setting where, typically, we have data and :

1. we fix a set of predictors (linear predictors for linear regression for instance).
2. we learn a good predictor from this set (by using the least-square method for instance).

Informally, *Probably Approximately Correct* Bayes bounds, or PAC-Bayes bounds, will allow us to define and study *randomized* or *aggregated* predictors. To do so, we will replace 2. by either $2'$. defining weights on the predictors and make a majority vote using them, or by $2''$. drawing a predictor from a yet to be chosen probability distribution.

2.6.1 Supervised learning setting

More formally, let us recall the supervised setting, where we rely on :

- data, or observations : $((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$ are assumed i.i.d samples from a probability distribution p on $(\mathcal{X} \times \mathcal{Y})$.
- a predictor, that is a measurable function $f : \mathcal{X} \to \mathcal{Y}$.
- a set of predictors that is fixed, indexed by a parameter set $\Theta : \{f_{\boldsymbol{\theta}}, \boldsymbol{\theta} \in \Theta\}$. For instance, in linear regression, $f_{\boldsymbol{\theta}}(x) = \boldsymbol{\theta}^\top \boldsymbol{x}$ for $\mathcal{X} = \Theta = \mathbb{R}^d$.
- a loss function, which is measurable $\ell : \mathcal{Y}^2 \to [0, +\infty)$ with $\ell(y, y) = 0$. In this section, we will assume it **bounded,** i.e. $0 \leq \ell \leq C$ even if recent works don't make this assumption [Haddouche, 2021].
- the generalization error/risk of a predictor (keeping only the dependency on f) :

$$\mathcal{R}(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim p}[\ell(f(\boldsymbol{x}), y)].$$

  As we will only consider predictors in $\{f_{\boldsymbol{\theta}}, \boldsymbol{\theta} \in \Theta\}$, we will write $\mathcal{R}(\boldsymbol{\theta}) := \mathcal{R}(f_{\boldsymbol{\theta}})$. This function is not accessible as it depends on the unknown probability distribution p.

- the empirical risk :

$$\hat{\mathcal{R}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i) := \frac{1}{n}\sum_{i=1}^{n} \ell_i(\boldsymbol{\theta}),$$

  which satisfies : $\mathbb{E}_{\mathcal{S}}[\hat{\mathcal{R}}(\boldsymbol{\theta})] = \mathcal{R}(\boldsymbol{\theta})$ where $\mathcal{S} = [(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)]$.
- an estimator that is a function : $\hat{\boldsymbol{\theta}} : \cup_{n=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^n \to \Theta$. The most famous example is the Empirical Risk Minimizer (ERM) :

$$\hat{\boldsymbol{\theta}}_{\text{ERM}} = \arg\min_{\boldsymbol{\theta}\in\Theta} \hat{\mathcal{R}}(\boldsymbol{\theta}).$$

2.6.2 PAC bounds

Ultimately, we would like to minimize $\mathcal{R}$ and not $\hat{\mathcal{R}}$, but we have only access to the latter. The ERM strategy is therefore motivated by the hope that these two functions are not too different, such that minimizing $\hat{\mathcal{R}}$ induces a minimization of $\mathcal{R}$. The following simple PAC bound provides a first guarantee :

Theorem 2.6.1. *Assume that* $\text{card}(\Theta) = M < +\infty$. *For any* $\varepsilon \in (0, 1)$,

$$\mathbb{P}_{\mathcal{S}}\left(\forall \boldsymbol{\theta} \in \Theta, \mathcal{R}(\boldsymbol{\theta}) \leq \hat{\mathcal{R}}(\boldsymbol{\theta}) + C\sqrt{\frac{\log \frac{M}{\varepsilon}}{2n}}\right) \geq 1 - \varepsilon.$$

This Theorem motivates the Empirical risk Minimization, as the ERM will satisfy :

$$\mathbb{P}_{\mathcal{S}}\left(\mathcal{R}(\hat{\boldsymbol{\theta}}_{\mathrm{ERM}}) \leq \inf_{\boldsymbol{\theta}\in\Theta}\left[\hat{\mathcal{R}}(\boldsymbol{\theta}) + C\sqrt{\frac{\log\frac{M}{\varepsilon}}{2n}}\right]\right) \geq 1-\varepsilon.$$

This Theorem means that with large probability $\mathcal{R}(\hat{\boldsymbol{\theta}}_{\mathrm{ERM}})$ is approximately equal to the value that is minimized on the empirical risk. The goal of PAC bounds, is to quantify to which extent this is true. We can notice that this is correct only if $\log(M)/n$ is small, meaning that $M < \exp(n)$.

Proof. As the loss is bounded, we can apply Markov inequality on $\mathbb{P}_{\mathcal{S}}(\mathcal{R}(\boldsymbol{\theta}) - \hat{\mathcal{R}}(\boldsymbol{\theta}) > s)$ and then use Hoeffding's inequality on the exponential term. After a minimization over s, we need a bound that perform uniformly over all $\boldsymbol{\theta} \in \Theta$, which implies the use of **an union bound** over the M elements of Θ (since its cardinal is finite). This is the final bound. See [Alquier, 2021] for a more detailed proof. □

2.6.3 PAC-Bayes bounds

Informally, PAC-Bayes bounds are a generalization of the union bound argument, which will allow to deal with any parameter sets Θ : finite or infinite, continuous... As a consequence, we will have to change the notion of an estimator.

Definition 2.6.2. *Let $\mathcal{P}(\Theta)$ be the set of all probability distributions on Θ. A data-dependent probability measure is a function :*

$$\hat{\rho} : \bigcup_{n=1}^{\infty} (\mathcal{X}\times\mathcal{Y})^n \to \mathcal{P}(\Theta),$$

with suitable measurability conditions.

This will change how we will handle our predictors, as now they are defined over a data-dependent probability distribution $\hat{\rho}$. To build the predictor, we can draw a random parameter $\tilde{\boldsymbol{\theta}} \sim \hat{\rho}$. This yields a "randomized estimator". Then, we can incorporate it in an average of predictors, defining a new predictor : $f_{\hat{\rho}}(\cdot) = \mathbb{E}_{\boldsymbol{\theta}\sim\hat{\rho}}[f_{\boldsymbol{\theta}}(\cdot)]$ called the aggregated predictor with weights $\hat{\rho}$.
Therefore, with PAC-Bayes bounds, we will extend the union bound argument to infinite, uncountable sets Θ. But one may ask : what will become the $\log(M)$ term, coming from the union bound, of Theorem 2.6.1 ? In general, this term will be replaced by the Kullback-Leibler divergence between $\hat{\rho}$ and a fixed probability measure π on Θ.

Definition 2.6.3. *Given two probability measures μ and ν in $\mathcal{P}(\Theta)$. The Kullback-Leibler (KL) divergence between μ and ν is defined by :*

$$KL(\mu\|\nu) = \int \log\left(\frac{\mathrm{d}\mu}{\mathrm{d}\nu}(\boldsymbol{\theta})\right)\mathrm{d}\mu(\boldsymbol{\theta}) \in [0,+\infty],$$

if μ has a density $\frac{\mathrm{d}\mu}{\mathrm{d}\nu}$ with respect to ν, and $KL(\mu\|\nu) = +\infty$ otherwise.

In the following, we will fix a probability measure $\pi \in \mathcal{P}(\Theta)$, called the **prior**. The first PAC-Bayesian bound is Catoni's bound :

Theorem 2.6.4 (Catoni's bound [Catoni, 2003])**.** *For any $\lambda > 0$, for any $\varepsilon \in (0,1)$,*

$$\mathbb{P}_{\mathcal{S}}\left(\forall \rho \in \mathcal{P}(\Theta), \mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\mathcal{R}(\boldsymbol{\theta})] \leq \mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\hat{\mathcal{R}}(\boldsymbol{\theta})] + \frac{\lambda C^2}{8n} + \frac{KL(\rho\|\pi) + \log\frac{1}{\varepsilon}}{\lambda}\right) \geq 1-\varepsilon.$$

With such a bound, the goal is to find the distribution ρ that minimizes the right hand side, i.e. :

$$\hat{\rho}_\lambda = \underset{\rho\in\mathcal{P}(\Theta)}{\arg\min}\ \mathbb{E}_{\boldsymbol{\theta}\sim\rho}[\hat{\mathcal{R}}(\boldsymbol{\theta})] + \frac{KL(\rho\|\pi)}{\lambda}. \tag{2.44}$$

It turns out that we know the answer to this minimization problem :

Corollary 2.6.5. *The Gibbs posterior is the minimizer of Equation 2.44 and has the following expression :*

$$\hat{\rho}_\lambda(\mathrm{d}\boldsymbol{\theta}) = \frac{e^{-\lambda\hat{\mathcal{R}}(\boldsymbol{\theta})}\pi(\mathrm{d}\boldsymbol{\theta})}{\mathbb{E}_{\boldsymbol{\beta}\sim\pi}[e^{-\lambda\hat{\mathcal{R}}(\boldsymbol{\beta})}]}. \tag{2.45}$$

In practice, sampling from the Gibbs posterior is intractable, and we will therefore have to use other schemes to find a distribution ρ that minimizes this bound.
The goal of PAC-Bayesian bounds is to become more specific depending on our problem : what is our estimator ? A linear function ? A neural network ? Is our loss bounded ? What are the constraints of the parameter space Θ ? As we will see in Chapter 7, we can optimize this bound to find an optimal $\hat{\rho}$, while being theoretically grounded thanks the PAC-Bayesian framework.

2.7 (Sliced) Optimal transport

The first to formulate the Optimal Transport framework was [Monge, 1781], who aimed at solving the problem of moving earth from one place to another, with the *least effort*. Some issues of the Monge formulation were resolved centuries later in [Kantorovitch, 1958]. Mathematically, this boils down to move a probability mass from one distribution to another, with the least cost. This cost is represented by a *cost function* c, operating as follow : let $\mu \in \mathcal{P}(\mathcal{X})$ be the *source distribution* and $\nu \in \mathcal{P}(\mathcal{Y})$ the *target distribution* where we assume $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \mathbb{R}^d$ for simplicity (in general, they have to be Polish spaces). Then, $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+ \cup \{+\infty\}$ is the function that for any $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}$ return the cost of transporting $\boldsymbol{x}$ to $\boldsymbol{y}$. It is typically chosen to evaluate how different/far $\boldsymbol{x}$ and $\boldsymbol{y}$ are from each other, i.e. the smaller $c(\boldsymbol{x}, \boldsymbol{y})$, the closer $\boldsymbol{x}$ and $\boldsymbol{y}$ are.

2.7.1 Wasserstein distances

Wasserstein distances are distances for comparing two distributions by solving a transport problem. They have become more and more popular in generative modelling with for example Wasserstein Generative Adversarial Networks [Arjovsky, 2017] which learn the underlying data distribution, allowing for the generation of new samples.

In the following, we will assume, without loss of generality, that the compared distributions are supported on the same space, i.e. $\mathcal{X} = \mathcal{Y}$.

Definition 2.7.1 ([Peyré, 2019])**.** *Let $\mathcal{X}$ be a Polish space equipped with a distance c and $p \in [1, +\infty)$. The Wasserstein distance of order p is defined for any $\mu, \nu \in \mathcal{P}(\mathcal{X})$ as*

$$\mathrm{W}_p(\mu, \nu) = \left(\inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} c(\boldsymbol{x}, \boldsymbol{y})^p \mathrm{d}\pi(\boldsymbol{x}, \boldsymbol{y}) \right)^{\frac{1}{p}},$$

with $\Pi(\mu, \nu) = \{\pi \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$: for any measurable sets $\mathcal{A} \subset \mathcal{X}, \mathcal{B} \subset \mathcal{Y}, \pi(\mathcal{A} \times \mathcal{Y}) = \mu(\mathcal{A}), \pi(\mathcal{X} \times \mathcal{B}) = \nu(\mathcal{B})\}$ the set of transport plans.

Let us now define the set $\mathcal{P}_p(\mathcal{X})$ of probability measures on $\mathcal{X}$ with their p-th moment finite :

$$\mathcal{P}_p(\mathcal{X}) = \left\{ \mu \in \mathcal{P}(\mathcal{X}) \ : \ \int_{\mathcal{X}} c(\boldsymbol{x}_0, \boldsymbol{x})^p \mathrm{d}\mu(\boldsymbol{x}) < +\infty, \text{ for some } \boldsymbol{x}_0 \in \mathcal{X} \right\}.$$

Then W_p is a metric on $\mathcal{P}_p(\mathcal{X})$ [Villani, 2008].
In some cases, the Wasserstein distribution has an analytical formula. This is the case when both distributions are Gaussian (see [Peyré, 2019] for the mathematical expression) or univariate.

Proposition 2.7.2 (Wasserstein distance for univariate distributions)**.** *Let $\mu, \nu \in \mathcal{P}_p(\mathbb{R})$, and denote by F_μ^{-1} and F_ν^{-1} the quantile functions of μ and ν respectively. Then*

$$\mathrm{W}_p^p(\mu, \nu) = \int_0^1 |F_\mu^{-1}(t) - F_\nu^{-1}(t)|^p \mathrm{d}t. \tag{2.46}$$

Practical aspects. In practice, we only have access to empirical distributions, i.e. samples from the real distributions. These empirical distributions are defined as $\mu_n = \frac{1}{n}\sum_{i=1}^n \delta_{\boldsymbol{x}_i}$ and $\nu_n = \frac{1}{n}\sum_{i=1}^n \delta_{\boldsymbol{y}_i}$ with $\{\boldsymbol{x}_i\}_{i=1}^n \sim \mu$ and $\{\boldsymbol{y}_i\}_{i=1}^n \sim \nu$, and $\delta_{\boldsymbol{z}}$ the Dirac distribution centered in $\boldsymbol{z}$.
In the case of univariate empirical distributions, i.e. $\mu_n, \nu_n \in \mathcal{P}_p(\mathbb{R})$, the Wasserstein distance has the following expression :

$$\mathrm{W}_p^p(\mu_n, \nu_n) = \frac{1}{n}\sum_{i=1}^n |x_{(i)} - y_{(i)}|^p. \tag{2.47}$$

where $x_{(1)} \leq ... \leq x_{(n)}$ and $y_{(1)} \leq ... \leq y_{(n)}$ are sorted samples.
In the univariate case, the problem is easy to solve as the sorting step takes $O(n \log(n))$ of computational complexity. However, in the multi-dimensional case, computing the Wasserstein distances aims at finding an optimal transport plan π, which is a permutation matrix matching the samples of μ_n to the ones of ν_n. In practice, this problem is solved using linear programming, having a $O(n^3 \log(n))$ worst case computational complexity [Peyré, 2019]. This is why there is a need to make the computation of this distance much faster.
The first algorithm aiming at performing optimal transport in high-dimension at a reduced computational cost is the Sinkhorn algorithm [Cuturi, 2013], a form of regularized optimal transport which adds a entropy penalty term to the computation of the Wasserstein distance.

Another method is the computation of the Sliced-Wasserstein distance, which takes advantage of the analytical expression of the Wasserstein distance for univariate distributions.

2.7.2 Sliced-Wasserstein distance

The Sliced-Wasserstein distance was introduced in [Rabin, 2012; Bonneel, 2015] and takes advantage of the expression of the Wasserstein distance for univariate distributions of Equation 2.46, which has a appealing computational cost. Informally, we will project the data distribution into one dimension with a random projection (or slice), yielding an univariate distribution. We will then use the formula of the Wasserstein distance between univariate distributions and average this over many slices. A more formal definition is :

Definition 2.7.3 (Sliced-Wasserstein distance (SW))**.** *Let $\mathcal{X} \in \mathbb{R}^d$, $p \in [1, +\infty)$ and denote the d-dimensional unit sphere by $\mathbb{S}^{d-1} = \{\boldsymbol{\theta} \in \mathbb{R}^d : \|\boldsymbol{\theta}\|_2 = 1\}$. For any $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$, denote by $\theta^\star : \mathcal{X} \to \mathbb{R}$ the linear form given by $\theta^\star(\boldsymbol{x}) = \langle \boldsymbol{\theta}, \boldsymbol{x} \rangle$. The Sliced-Wasserstein distance of order p is defined for any $\mu, \nu \in \mathcal{P}_p(\mathcal{X})$ as*

$$\mathrm{SW}_p^p(\mu, \nu) = \int_{\mathbb{S}^{d-1}} \mathrm{W}_p^p(\theta^\star_\sharp \mu, \theta^\star_\sharp \nu) \mathrm{d}\sigma(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta} \sim \sigma}[\mathrm{W}_p^p(\theta^\star_\sharp \mu, \theta^\star_\sharp \nu)], \tag{2.48}$$

where σ is the uniform distribution on $\mathbb{S}^{d-1}$, and for any $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$, $\theta^\star_\sharp = (\theta^\star)_\sharp$ denotes the push-forward operator associated to $\theta^\star$, i.e. $\theta_\sharp \mu(\mathcal{A}) = \mu(\theta^{-1}(\mathcal{A}))$ with $\mu^{-1}(\mathcal{A}) = \{\boldsymbol{x} \in \mathcal{X} : \theta(\boldsymbol{x}) \in \mathcal{A}\}$.

The Sliced-Wasserstein distance measures the dissimilarity between μ and ν by projecting them into one dimension using the slices $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ and averaging theses dissimilarities over all slices. In practice, we perform a Monte-Carlo sampling of the integral over $\mathbb{S}^{d-1}$ to approximate the distance : we will draw $L \in \mathbb{N}^*$ samples i.i.d. from σ, denoted by $\{\boldsymbol{\theta}_l\}_{l=1}^L$:

$$\mathrm{SW}_{p,L}^p(\mu, \nu) = \frac{1}{L} \sum_{l=1}^{L} \mathrm{W}_p^p(\theta^\star_{l\sharp} \mu, \theta^\star_{l\sharp} \nu) \tag{2.49}$$

where here the Wasserstein distance has an analytical solution of Equation 2.46.
With real data, we only have access to empirical distributions, leading to the following expression :

$$\mathrm{SW}_{p,L}^p(\mu_n, \nu_n) = \frac{1}{L} \frac{1}{n} \sum_{l=1}^{L} \sum_{i=1}^{n} |\langle \boldsymbol{\theta}_l, \boldsymbol{x}_{\gamma_l(i)} \rangle - \langle \boldsymbol{\theta}_l, \boldsymbol{y}_{\tau_l(i)} \rangle|^p \tag{2.50}$$

where γ_l and τ_l are the sorting permutations of the points projected by $\boldsymbol{\theta}_l$. This approximation methods benefits from a reduced computational complexity of $O(Ldn + Ln\log(n))$ compared to the Wasserstein distance thanks to the simple projecting and sorting operations.

Chapter 3

Reservoir Computing meets Recurrent Kernels and Structured Transforms

This chapter is based on [Dong, 2020].

Reservoir Computing is a class of simple yet efficient Recurrent Neural Networks where internal weights are fixed at random and only a linear output layer is trained. In the large size limit, such random neural networks have a deep connection with kernel methods. Our contributions are threefold : a) We rigorously establish the recurrent kernel limit of Reservoir Computing and prove its convergence. b) We test our models on chaotic time series prediction, a classic but challenging benchmark in Reservoir Computing, and show how the Recurrent Kernel is competitive and computationally efficient when the number of data points remains moderate. c) When the number of samples is too large, we leverage the success of structured Random Features for kernel approximation by introducing Structured Reservoir Computing. The two proposed methods, Recurrent Kernel and Structured Reservoir Computing, turn out to be much faster and more memory-efficient than conventional Reservoir Computing.

3.1 Introduction

Understanding Neural networks in general, and how to train Recurrent Neural Networks (RNNs) in particular, remains a central question in modern machine learning. Indeed, backpropagation in recurrent architectures faces the problem of exploding or vanishing gradients [Pascanu, 2013 ; Salehinejad, 2017], reducing the effectiveness of gradient-based optimization algorithms. While there exist very powerful and complex RNNs for modern machine learning tasks, interesting questions still remain regarding simpler ones. In particular, Reservoir Computing (RC) is a class of simple but efficient Recurrent Neural Networks introduced in [Jaeger, 2001] with the Echo-State Network, where internal weights are fixed randomly and only a last linear layer is trained [Verstraeten, 2007]. As the training reduces to a well-understood linear regression, Reservoir

Computing enables us to investigate separately the complexity of neuron activations in RNNs. With a few hyperparameters, we can tune the dynamics of the reservoir from stable to chaotic and performances are increased when RC operates close to the chaotic regime [Lukoševičius, 2009]. Despite its simplicity, Reservoir Computing is not fully efficient : computational and memory costs grow quadratically with the number of neurons. To tackle this issue, efficient computation schemes have been proposed based on sparse weight matrices [Lukoševičius, 2009]. Moreover, there is an active community developing novel hardware solutions for energy-efficient, low-latency RC [Pathak, 2018]. Based on dedicated electronics [Antonik, 2015 ; Wang, 2015 ; Zhang, 2015 ; Jin, 2017], optical computing [Larger, 2012 ; Duport, 2012 ; Van der Sande, Dong, 2018 ; Dong, 2019], or other original physical designs [Tanaka, 2019], they leverage the robustness and flexibility of RC. Reservoir Computing has already been used in a variety of tasks, such as speech recognition and robotics [Lukoševičius, 2012] but also combined with Random Convolutional Neural Networks for image recognition [Tong, 2018] and Reinforcement Learning [Chang, 2020]. A very promising application today is chaotic time series prediction, where the RC dynamics close to chaos may prove a very important asset [Pathak, 2018]. Reservoir Computing also represents an important model in computational neuroscience, as parallels can be drawn with specific regions of the brain behaving like a set of randomly-connected neurons [Hinaut, 2013].
As RC embeds input data in a high-dimensional reservoir, it has already been linked with kernel methods [Lukoševičius, 2009], but merely as an interesting interpretation for discussion. In our opinion, this point of view has not been exploited to its full potential yet. A study derived the explicit formula of the corresponding recurrent kernel associated with RC [Hermans, 2012], this important result meaning the infinite-width limit of RC is a deterministic Recurrent Kernel (RK). Still, no theoretical study of convergence towards this limit has been conducted previously and the computational complexity of Recurrent Kernels has not been derived yet.
In this work, we draw the link between RC and the rich literature on Random Features for kernel approximation [Rahimi, 2008 ; Rahimi, 2009 ; Rudi, 2017b ; Carratino, 2018 ; Liu, 2020] . To accelerate and scale-up the computation of Random Features, one can use optical implementations [Saade, 2016b ; Ohana, 2020] or structured transforms [Le, 2013 ; Yu, 2016], providing a very efficient method for kernel approximation. Structured transforms such as the Fourier or Hadamard transforms can be computed in $O(n \log n)$ complexity and, coupled with random diagonal matrices, they can replace the dense random matrix used in Random Features.
Finally, we note that Reservoir Computing can be unrolled through time and interpreted as a multilayer perceptron. The theoretical study of such randomized neural networks through the lens of kernel methods has attracted a lot of attention recently [Jacot, 2018 ; Mei, 2019 ; Gallicchio, 2020], which provides a further motivation to our work. Some parallels were already drawn between Recurrent Neural Networks and kernel methods [Liang, 2019 ; Chen, 2019], but they do not tackle the high-dimensional random case of Reservoir Computing.

Main contributions — In this chapter, our goal is to bridge the gap between the considerable amount of results on kernels methods, random features — structured or not — and Reservoir Computing.

First, we rigorously prove the convergence of Reservoir Computing towards Recurrent Kernels provided standard assumptions and derive convergence rates in $O(1/\sqrt{N})$, with N being the number of neurons. We then numerically show convergence is achieved in a large variety of cases and does not occur in practice only when the activation function is unbounded (for instance with ReLU).
When the number of training points is large, the complexity of RK grows ; this is a common drawback of kernel methods. To circumvent this issue, we propose to accelerate conventional Reservoir Computing by replacing the dense random weight matrix with a structured transform. In practice, Structured Reservoir Computing (SRC) allows to scale to very large reservoir sizes easily, as it is faster and more memory-efficient than conventional Reservoir Computing, without compromising performance.
These techniques are tested on chaotic time series prediction, and they all present comparable results in the large-dimensional setting. We also derive the computational complexities of each algorithm and detail how Recurrent Kernels can be implemented efficiently. In the end, the two acceleration techniques we propose are faster than Reservoir Computing and can tackle equally complex tasks. A public repository is available at https://github.com/rubenohana/Reservoir-computing-kernels.

3.2 Recurrent Kernels and Structured Reservoir Computing

Here, we briefly describe the main concepts used in this paper. We recall the definition of Reservoir Computing and Random Features, define Recurrent Kernels (RKs) and introduce Structured Reservoir Computing (SRC).
Reservoir Computing (RC) as a Recurrent Neural Network receives a sequential input $\boldsymbol{i}^{(t)} \in \mathbb{R}^d$, for $t \in \mathbb{N}$. We denote by $\boldsymbol{x}^{(t)} \in \mathbb{R}^N$ the state of the reservoir, N being the number of neurons in the reservoir. Its dynamics is given by the following recurrent equation :

$$\boldsymbol{x}^{(t+1)} = \frac{1}{\sqrt{N}} f\left(\mathbf{W_r}\, \boldsymbol{x}^{(t)} + \mathbf{W_i}\, \boldsymbol{i}^{(t)}\right), \tag{3.1}$$

where $\mathbf{W_r} \in \mathbb{R}^{N\times N}$ and $\mathbf{W_i} \in \mathbb{R}^{N\times d}$ are respectively the reservoir and input weight matrices. They are fixed and random : each weight is drawn according to an i.i.d. Gaussian distribution with variances σ_r^2 and σ_i^2, respectively. Finally, f is an element-wise non-linearity, typically a hyperbolic tangent. To refine the control of the reservoir dynamics, it is possible to add a random bias and a leak rate. In the following, we will keep the minimal formalism of Equation 3.1 for conciseness.
We use the reservoir to learn how to predict a given output $\boldsymbol{o}^{(t)} \in \mathbb{R}^c$ for example. The output predicted by the network $\hat{\boldsymbol{o}}^{(t)}$ is obtained after a final layer :

$$\hat{\boldsymbol{o}}^{(t)} = \mathbf{W_o}\, \boldsymbol{x}^{(t)}. \tag{3.2}$$

Since only these output weights $\mathbf{W_o} \in \mathbb{R}^{c\times N}$ are trained, the optimization problem boils down to linear regression. Training is typically not a limiting factor in RC, in sharp contrast with other

neural network architectures. The expressivity and power of Reservoir Computing rather lies in the high-dimensional non-linear dynamics of the reservoir.

Kernel methods are non-parametric approaches to learning. Essentially, a kernel is a function measuring a correlation between two points $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^p$. A specificity of kernels is that they can be expressed as the inner product of feature maps $\boldsymbol{\varphi} : \mathbb{R}^p \to \mathcal{H}$ in a possibly infinite-dimensional Hilbert space $\mathcal{H}$, i.e. $k(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{\varphi}(\boldsymbol{u}), \boldsymbol{\varphi}(\boldsymbol{v}) \rangle_{\mathcal{H}}$. Kernel methods enable the use of linear methods in the non-linear feature space $\mathcal{H}$. Famous examples of kernel functions are the Gaussian kernel $k(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{v}\|^2}{2\sigma^2}\right)$ or the arcsine kernel $k(\boldsymbol{u}, \boldsymbol{v}) = \frac{2}{\pi} \arcsin \frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|}$. When the dataset becomes large, it is expensive to numerically compute the kernels between all pairs of data points.

Random Features have been developed in [Rahimi, 2008] to overcome this issue. This celebrated technique introduces a random mapping $\boldsymbol{\phi} : \mathbb{R}^p \to \mathbb{R}^N$ such that the kernel is approximated in expectation :

$$k(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{\varphi}(\boldsymbol{u}), \boldsymbol{\varphi}(\boldsymbol{v}) \rangle_{\mathcal{H}} \approx \langle \boldsymbol{\phi}(\boldsymbol{u}), \boldsymbol{\phi}(\boldsymbol{v}) \rangle_{\mathbb{R}^N}, \tag{3.3}$$

with $\boldsymbol{\phi}(\boldsymbol{u}) = \frac{1}{\sqrt{N}}[f(\langle \boldsymbol{w}_1, \boldsymbol{u} \rangle), ..., f(\langle \boldsymbol{w}_N, \boldsymbol{u} \rangle)]^\top \in \mathbb{R}^N$ and random vectors $\boldsymbol{w}_1, ..., \boldsymbol{w}_N \in \mathbb{R}^p$. Depending on f and the distribution of $\{\boldsymbol{w}_i\}_{i=1}^N$, we can approximate different kernel functions.

There are two major classes of kernel functions : translation-invariant **(TI)** kernels and rotation-invariant **(RI)** kernels. In our study, we will consider TI kernels of the form $k(\boldsymbol{u}, \boldsymbol{v}) = k(\|\boldsymbol{u} - \boldsymbol{v}\|_2^2)$ and RI kernels of the form $k(\boldsymbol{u}, \boldsymbol{v}) = k(\langle \boldsymbol{u}, \boldsymbol{v} \rangle)$. Both can be approximated using Random Features [Rahimi, 2008 ; Kar, 2012]. For example, Random Fourier Features (RFFs) defined by :

$$\boldsymbol{\phi}(\boldsymbol{u}) = \frac{1}{\sqrt{N}}[\cos(\langle \boldsymbol{w}_1, \boldsymbol{u} \rangle), ..., \cos(\langle \boldsymbol{w}_N, \boldsymbol{u} \rangle), \sin(\langle \boldsymbol{w}_1, \boldsymbol{u} \rangle), ..., \sin(\langle \boldsymbol{w}_N, \boldsymbol{u} \rangle)]^\top, \tag{3.4}$$

approximate any TI kernel (provided $k(0) = 1$). For example, when $\boldsymbol{w}_1, ..., \boldsymbol{w}_N \sim \mathcal{N}(0, \sigma^{-2}\mathbf{I}_p)$, we approximate the Gaussian kernel. A detailed taxonomy of Random Features can be found in [Liao, 2018] and more details about kernel methods and random features can be found in Section 2.2.

Random Features can be more computationally efficient than kernel methods, when their number N is smaller than the number of data points n. For this particular reason, Random Features are a method of choice to implement large-scale kernel-based methods.

Link with Reservoir Computing. It is straightforward to notice that reservoir iterations of Equation 3.1 can be interpreted as a Random Feature embedding of a vector $[\boldsymbol{x}^{(t)}, \boldsymbol{i}^{(t)}]$ (of dimension $p = N + d$), multiplied by $\mathbf{W} = [\mathbf{W_r}, \mathbf{W_i}]$. This means the inner product between two reservoirs $\boldsymbol{x}^{(t)}$, $\boldsymbol{y}^{(t)}$ driven respectively by two inputs $\boldsymbol{i}^{(t)}$ and $\boldsymbol{j}^{(t)}$ converges to a deterministic kernel as N tends to infinity :

$$\langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)} \rangle \approx k([\boldsymbol{x}^{(t)}, \boldsymbol{i}^{(t)}], [\boldsymbol{y}^{(t)}, \boldsymbol{j}^{(t)}]). \tag{3.5}$$

As explained previously, this kernel depends on the choice of f and the distribution of $\mathbf{W_r}$ and $\mathbf{W_i}$.

By denoting $l^{(t)} = \sigma_i^2 \langle \boldsymbol{i}^{(t)}, \boldsymbol{j}^{(t)} \rangle$ and $\Delta^{(t)} = \sigma_i^2 \|\boldsymbol{i}^{(t)} - \boldsymbol{j}^{(t)}\|^2$, TI and RI kernels are then of the

form :

$$k([\boldsymbol{x}^{(t)}, \boldsymbol{i}^{(t)}], [\boldsymbol{y}^{(t)}, \boldsymbol{j}^{(t)}]) = k(\sigma_r^2 \langle \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)} \rangle + l^{(t)}) \quad \text{(RI)} \tag{3.6}$$

$$= k(\sigma_r^2 \|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\|^2 + \Delta^{(t)}) \quad \text{(TI)}. \tag{3.7}$$

The Recurrent Kernel limit. Looking at Equations 3.6 and 3.7, we notice the kernel at time t depends on approximations of kernels at previous times in a recursive manner. Here, we introduce Recurrent Kernels to remove the dependence in $\boldsymbol{x}^{(t)}$ and $\boldsymbol{y}^{(t)}$.
We suppose for the sake of simplicity $\boldsymbol{x}^{(0)} = \boldsymbol{y}^{(0)} = 0$. We define RI recurrent kernels as :

$$\begin{cases} k_1\left(l^{(0)}\right) = k\left(l^{(0)}\right) \\ k_{t+1}\left(l^{(t)}, ..., l^{(0)}\right) = k\left(\sigma_r^2 k_t\left(l^{(t-1)}, ..., l^{(0)}\right) + l^{(t)}\right), \quad \text{for } t \in \mathbb{N}^*. \end{cases} \tag{3.8}$$

Similarly for TI recurrent kernels with Random Fourier Features, exploiting the property in Equation 3.4 that $\|\boldsymbol{x}^{(t)}\|^2 = \|\boldsymbol{y}^{(t)}\|^2 = 1$:

$$\begin{cases} k_1\left(\Delta^{(0)}\right) = k\left(\Delta^{(0)}\right) \\ k_{t+1}\left(\Delta^{(t)}, ..., \Delta^{(0)}\right) = k\left(\sigma_r^2\left(2 - 2k_t\left(\Delta^{(t-1)}, ..., \Delta^{(0)}\right)\right) + \Delta^{(t)}\right), \quad \text{for } t \in \mathbb{N}^*. \end{cases} \tag{3.9}$$

These Recurrent Kernel definitions describe hypothetical asymptotic limits of large-dimensional Reservoir Computing, interpreted as recurrent Random Features. We will study in Section 3.3.1 the convergence towards this limit.
Structured Reservoir Computing. In the Random Features literature, it is common to use structured transforms to speed-up computations of random matrix multiplications [Le, 2013 ; Yu, 2016]. They have also been introduced for trained architectures, with Deep [Moczulski, 2015] and Recurrent Neural Networks [Arjovsky, 2016].
We propose to replace the dense random weight matrices $\mathbf{W} = [\mathbf{W_r}, \mathbf{W_i}]$ by a succession of Hadamard matrices $\mathbf{H}$ (structured orthonormal matrices composed of $\pm 1/\sqrt{p}$ components) and diagonal random matrices $\mathbf{D}_i$ for $i = 1, 2, 3$ sampled from an i.i.d. Rademacher distribution [Yu, 2016] :

$$\mathbf{W} = \frac{\sqrt{p}}{\sigma} \mathbf{H}\mathbf{D}_1\mathbf{H}\mathbf{D}_2\mathbf{H}\mathbf{D}_3. \tag{3.10}$$

We use the Hadamard transform for its simplicity and the availability of high-performance libraries in [Thomas, 2018]. This structured transform provides the two main properties of a dense random matrix : mixing the activations of neurons (Hadamard transform) and randomness (diagonal matrices).

3.3 Convergence theorem and computational complexity

3.3.1 Convergence rates

Our first main result is a convergence theorem of Reservoir Computing to its kernel limit. We use Bernstein's concentration inequality in our recurrent setting. Several assumptions will be

necessary :

- The kernel function k is Lipschitz-continuous with constant L, i.e. $|k(a) - k(b)| \leq L|a - b|$.
- The random matrices $\mathbf{W_r}$ and $\mathbf{W_i}$ are resampled for each t to obtain uncorrelated reservoir updates : $\boldsymbol{x}^{(t+1)} = \frac{1}{\sqrt{N}} f(\mathbf{W_r}^{(t)}\boldsymbol{x}^{(t)} + \mathbf{W_i}^{(t)}\boldsymbol{i}^{(t)})$. This assumption is required for our theoretical proof of convergence, but we show convergence is reached numerically even without redrawing the weight matrices, which is standard in Reservoir Computing (in Figure 3.1).
- The function f is bounded by a constant κ almost surely, i.e. $|f(\mathbf{W_r}^{(t)}\boldsymbol{x}^{(t)} + \mathbf{W_i}^{(t)}\boldsymbol{i}^{(t)})| \leq \kappa$.

Theorem 3.3.1. *(Rotation-invariant kernels) For the RI recurrent kernel defined in Equation 3.8, under the assumptions detailed above, and with $\Lambda = \sigma_r^2 L$. For all $t \in \mathbb{N}$, the following inequality is satisfied for any $\delta > 0$ with probability at least $1 - 2(t+1)\delta$:*

$$\left|\langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)} \rangle - k_{t+1}(l^{(t)}, ..., l^{(0)})\right| \leq \frac{1 - \Lambda^{t+1}}{1 - \Lambda}\Theta(N) \qquad \text{if} \quad \Lambda \neq 1 \tag{3.11}$$

$$\leq (t+1)\Theta(N) \qquad \text{if} \quad \Lambda = 1 \tag{3.12}$$

with $\Theta(N) = \frac{4\kappa^2 \log\frac{1}{\delta}}{3N} + 2\kappa^2\sqrt{\frac{2\log\frac{1}{\delta}}{N}}$.

Proof. We use the following Proposition (Theorem 3 of [Boucheron, 2003] restated in Proposition 1 of [Rudi, 2017b]) :

Proposition 3.3.2. *(Bernstein inequality for a sum of random variables). Let $X_1, ..., X_N$ be a sequence of i.i.d. random variables on $\mathbb{R}$ with zero mean. If there exist $R, S \in \mathbb{R}$ such that $|X_i| \leq R$ almost everywhere and $\mathbb{E}[X_i^2] \leq S$ for $i \in \{1, ..., N\}$, then for any $\delta > 0$ the following holds with probability at least $1 - 2\delta$:*

$$\left|\frac{1}{N}\sum_{i=1}^{N} X_i\right| \leq \frac{2R\log\frac{1}{\delta}}{3N} + \sqrt{\frac{2S\log\frac{1}{\delta}}{N}}. \tag{3.13}$$

Under the assumptions, Proposition 3.3.2 yields with probability greater than $1 - 2\delta$:

$$\left|\langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)} \rangle - k([\boldsymbol{x}^{(t)}, \boldsymbol{i}^{(t)}], [\boldsymbol{y}^{(t)}, \boldsymbol{j}^{(t)}])\right| \leq \frac{4\kappa^2 \log\frac{1}{\delta}}{3N} + 2\kappa^2\sqrt{\frac{2\log\frac{1}{\delta}}{N}} = \Theta(N). \tag{3.14}$$

It means the larger the reservoir, the more Random Features N we sample, and the more the inner product of reservoir states concentrates towards its expectation value, at a rate $O(1/\sqrt{N})$. We now apply this inequality recursively to complete the proof, based on the observation that both Equations 3.11 and 3.12 are equivalent to : $\left|\langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)} \rangle - k_{t+1}(l^{(t)}, ..., l^{(0)})\right| \leq (1 + \Lambda + \Lambda^2 + ... + \Lambda^t)\Theta(N)$.
For $t = 0$, provided $\boldsymbol{x}^{(0)} = \boldsymbol{y}^{(0)} = 0$, we have, according to Equation 3.14, with probability at least $1 - 2\delta$:

$$\left|\langle \boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)} \rangle - k_1(l^{(0)})\right| \leq \Theta(N). \tag{3.15}$$

For any time $t \in \mathbb{N}^*$, let us assume the following event A_t is true with probability $\mathbb{P}(A_t) \geq 1-2t\delta$:

$$\left|\langle \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)} \rangle - k_t(l^{(t-1)}, ..., l^{(0)})\right| \leq (1 + \ldots + \Lambda^{t-1})\Theta(N). \tag{3.16}$$

Using the Lipschitz-continuity of k, this inequality is equivalent to :

$$\left|k(\sigma_r^2\langle \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)} \rangle + l^{(t)}) - k(\sigma_r^2 k_t(l^{(t-1)}, ..., l^{(0)}) + l^{(t)})\right| \leq (\Lambda + \ldots + \Lambda^t)\Theta(N). \tag{3.17}$$

With Equation 3.14, the following event B_t is true with probability $\mathbb{P}(B_t) \geq 1 - 2\delta$:

$$\left|\langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)} \rangle - k(\sigma_r^2\langle \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)} \rangle + l^{(t)})\right| \leq \Theta(N). \tag{3.18}$$

Summing Equations 3.17 and 3.18, with the triangular inequality and a union bound, the following event A_{t+1} is true with probability $\mathbb{P}(A_{t+1}) \geq \mathbb{P}(B_t \cap A_t) = \mathbb{P}(B_t) + \mathbb{P}(A_t) - \mathbb{P}(B_t \cup A_t) \geq 1 - 2\delta + 1 - 2t\delta - 1 \geq 1 - 2(t+1)\delta$:

$$\left|\langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)} \rangle - k_{t+1}(l^{(t)}, ..., l^{(0)})\right| \leq (1 + \ldots + \Lambda^t)\Theta(N). \tag{3.19}$$

□

A statement and proof of a similar convergence bound for TI recurrent kernels is provided in Appendix 3.6.1.

3.3.2 Numerical study of convergence

The previous theoretical study required three important assumptions that may not be valid for Reservoir Computing in practice. Moreover, there is still no rigorous proof on the convergence of Structured Random Features in the non-recurrent case due to the difficulty to deal with correlations between them. Thus, we numerically investigate whether convergence of RC and SRC towards the Recurrent Kernel limit is achieved in practice.

In Figure 3.1, we numerically compute the Mean-Squared Error (MSE) between the inner products obtained with a Recurrent Kernel and RC/SRC for different number of neurons in the reservoir. We generate 50 i.i.d. Gaussian input time series $\boldsymbol{i}_k^{(t)}$ of length T, for $k = 1, \ldots, 50$ and $t = 0, \ldots, T-1$. Each time series is fed into 50 reservoirs that share the same random weights, for RC and SRC. We compute the MSE between inner products of pairs of final reservoir states $\langle \boldsymbol{x}_k^{(T)}, \boldsymbol{x}_l^{(T)} \rangle$ and the deterministic limit obtained directly with $k_T(\boldsymbol{i}_k^{(T-1)}, \boldsymbol{i}_l^{(T-1)}, \ldots, \boldsymbol{i}_k^{(0)}, \boldsymbol{i}_l^{(0)})$, for all $k, l = 1, \ldots, 50$. The computation is vectorized to be efficiently implemented on a GPU. Three different activation functions, the rectified linear unit (ReLU), the error function (Erf), and Random Fourier Features defined in Equation 3.4, have been tested with different variances of the reservoir weights. The larger the reservoir weights, the more unstable the reservoir dynamics becomes.

Nonetheless, convergence is achieved in a large variety of settings, even when the assumptions of the previous theorem are not satisfied. For example, the ReLU non-linearity is not bounded and converges when $\sigma_r^2 \geq 1$. It is interesting to notice even for a large variance $\sigma_r^2 = 4$ do Reservoir

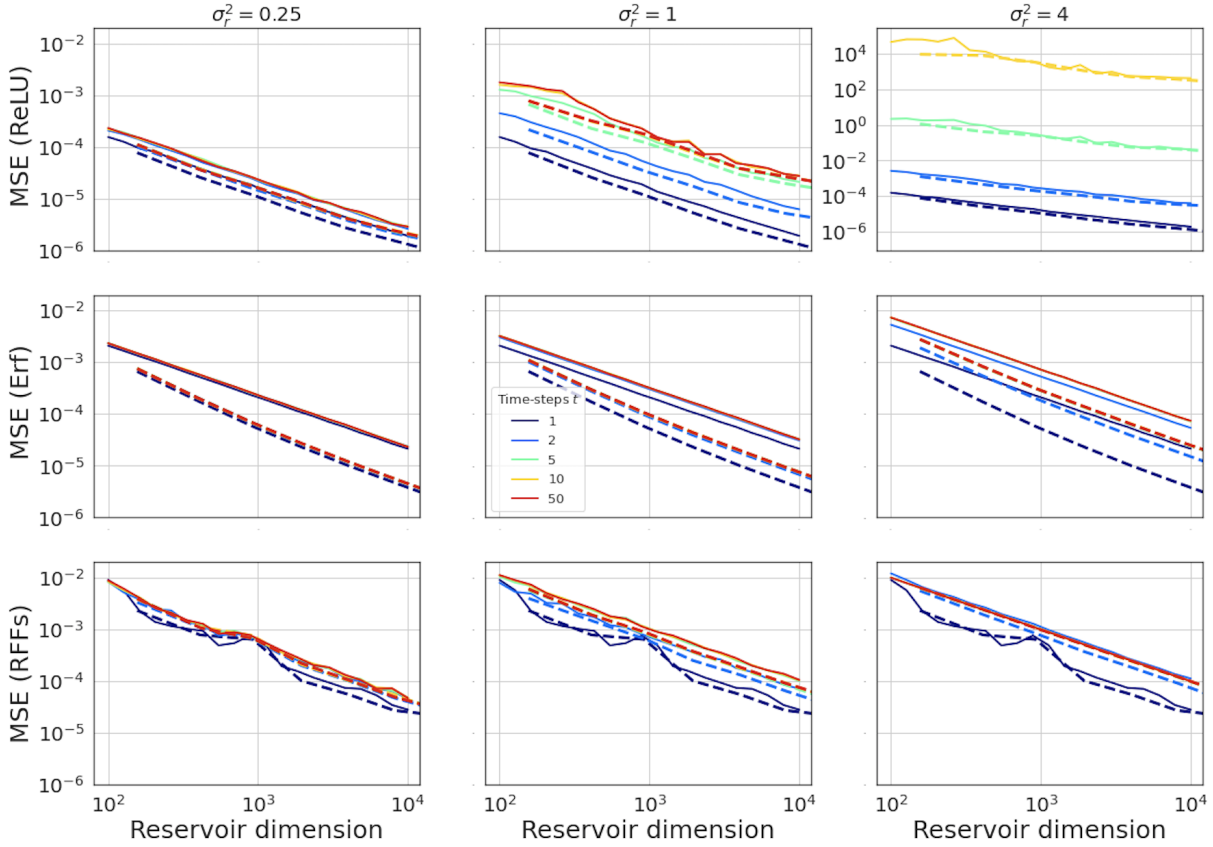


FIGURE 3.1 – Convergence of Reservoir Computing towards its Recurrent Kernel limit for different variances of the reservoir weights σ_r^2 (columns), activation functions (lines : ReLU, Erf, RFFs) and times, for RC **(solid lines)** and SRC **(dashed lines)**. We observe that for the two bounded activation functions (Erf and RFFs), RC always converge towards the RK limit even at large times t. For ReLU, RC converges when $\sigma_r^2 = 0.25$ and 1, and diverges as t increases when $\sigma_r^2 = 4$. We also observe that SRC always yields equal or faster convergence than RC. The MSE decreases with an $O(1/N)$ scaling, which is consistent with the convergence rates derived in Theorem 3.3.1.

Computing and Structured Reservoir Computing converge towards the RK limit for the second and third activation functions. This behavior has been consistently observed with any bounded f.

On the other hand, Structured Reservoir Computing seems to always converge faster than Reservoir Computing. We thus confirm in the recurrent case the intriguing effectiveness of Structured Random Features [Choromanski, 2017b], that may originate from the orthogonality of the matrix $\mathbf{W_r}$ in SRC.

As a final remark, weight matrices in Figure 3.1 were not redrawn as supposed in Section 3.3.1. This assumption was necessary as correlations are often difficult to take into account in a theoretical setting. This is important for Reservoir Computing as it would be unrealistically slow to draw new random matrices at each time step.

3.3.3 When to use Recurrent Kernels or Structured Reservoir Computing ?

The two proposed alternatives to Reservoir Computing, Recurrent Kernels and Structured Reservoir Computing, are computationally efficient. To understand which algorithm to use for chaotic system prediction, we need to focus on the limiting operation in the whole pipeline of Reservoir Computing, the recurrent iterations. They correspond to Equation 3.1 for RC/SRC and Equations 3.8 and 3.9 for RK. We have a time series of dimension d, that we split into train/test datasets of lengths n and m respectively. The exact computational and memory complexities of each step are described in Table 3.1.

Forward : In both Reservoir Computing and Structured Reservoir Computing, Equation 3.1 needs to be repeated as many times as the length of the time series. For Reservoir Computing, it requires a multiplication by a dense $N \times N$ matrix, the associated complexity scales as $O(N^2)$. On the other hand, Structured Reservoir Computing uses a succession of Hadamard and diagonal matrix multiplications, reducing the complexity per iteration to $O(N \log N)$.

Recurrent Kernels need to recurrently compute Equations 3.8 and 3.9 for all pairs of input points. For chaotic time series prediction, this corresponds to a $n \times n$ kernel matrix for training, and another kernel matrix of size $n \times m$ for testing. To keep computation manageable, we use a well-known property in Reservoir Computing, called the Echo-State Property : the reservoir state should not depend on the initialization of the network, i.e. the reservoir needs to have a finite memory τ. This property is important in Reservoir Computing and has been studied extensively [Jaeger, 2001 ; Schrauwen, 2009 ; Wainrib, 2016 ; Inubushi, 2017]. Transposed in the Recurrent Kernel setting, it means we can fix the number of iterations of Equations 3.8 and 3.9 to τ, by using a sliding window to construct shorter time series if necessary. A preliminary numerical study of the stability of Recurrent Kernels is presented in Appendix 3.6.4.

Training requires, after a forward pass on the training dataset, to solve an $n \times N$ linear system for RC/SRC and a $n \times n$ linear system for RK. It is important to note SRC and RK do not accelerate this linear training step. We will use Ridge Regression with regularization parameter α to learn $\mathbf{W_o}$.

Prediction in Reservoir Computing and Structured Reservoir Computing only requires the computation of reservoir states and multiplication by the learned output weights. Recurrent Kernels need to compute a new kernel matrix for every pair $(\boldsymbol{i}_r, \boldsymbol{j}_q)$ with $\boldsymbol{i}_r$ in the training set and $\boldsymbol{j}_q$ in the testing set. Note that the prediction step includes a forward pass on the test set, followed by a linear model.

	Reservoir Computing	Structured Reservoir Computing	Recurrent Kernel
Forward	$O(nN^2)$	$O(nN \log N)$	$O(n^2\tau)$
Training	$O(nN^2 + N^3)$	$O(nN^2 + N^3)$	$O(n^3)$
Prediction	$O(mN^2)$	$O(mN \log N)$	$O(mn\tau)$
Memory	$O(nN + N^2)$	$O(nN)$	$O(n^2 + mn)$

TABLE 3.1 – Computational and memory complexity of the three algorithms. SRC accelerates the forward pass and decreases memory complexity compared to conventional RC. The complexity of RK depends on the number of training and testing points and is advantageous when $n \ll N$.

Algorithm 2 Recurrent Kernel algorithm

Result : Predictions $\hat{\boldsymbol{o}}^{(t)} \in \mathbb{R}^{c\times m}$.
Input : A train set $\{\boldsymbol{i}_r^{(t)}\}_{r=1}^{n} \in \mathbb{R}^{\tau\times d}$ with outputs $\boldsymbol{o} \in \mathbb{R}^{c\times n}$, a test set $\{\boldsymbol{j}_q^{(t)}\}_{q=1}^{m} \in \mathbb{R}^{\tau\times d}$.
Training : Initialize an $n \times n$ kernel matrix $\boldsymbol{G}^{(0)} = 0$.
for $t = 0, \dots, \tau - 1$ **do**
  Compute $(\boldsymbol{G}^{(t+1)})_{rs} = k_{t+1}(\boldsymbol{i}_r^{(t)}, \boldsymbol{i}_s^{(t)}, \dots, \boldsymbol{i}_r^{(0)}, \boldsymbol{i}_s^{(0)})$ using Eq. (3.8) or (3.9) and $(\boldsymbol{G}^{(t)})_{rs}$.
end for
Compute the output weights $\mathbf{W_o} \in \mathbb{R}^{c\times n}$ that minimize $\|\boldsymbol{o} - \mathbf{W_o}\boldsymbol{G}^{(\tau)}\|_2^2 + \alpha\|\mathbf{W_o}\|_2^2$
Prediction : Initialize an $n \times m$ kernel matrix $\boldsymbol{K}^{(0)} = 0$.
for $t = 1, \dots, \tau$ **do**
  Compute $(\boldsymbol{K}^{(t+1)})_{rq} = k_{t+1}(\boldsymbol{i}_r^{(t)}, \boldsymbol{j}_q^{(t)}, \dots, \boldsymbol{i}_r^{(0)}, \boldsymbol{j}_q^{(0)})$ using Eq. (3.8) or (3.9) and $(\boldsymbol{K}^{(t)})_{rq}$.
end for
Compute the predicted outputs $\hat{\boldsymbol{o}}^{(t)} = \mathbf{W_o}\boldsymbol{K}^{(\tau)}$.

3.4 Chaotic time series prediction

Chaotic time series prediction is a task arising in many different fields such as fluid dynamics, financial or weather forecasts. By definition, it is difficult to predict their future evolution since initially small differences get amplified exponentially. Recurrent Neural Networks and in particular Reservoir Computing represent very powerful tools to solve this task [Antonik, 2018 ; Vlachas, 2019].

The Kuramoto-Sivashinsky (KS) chaotic system is defined by a fourth-order partial derivative equation in space and time [Kuramoto, 1978 ; Sivashinsky, 1977]. We use a discretized version from a publicly available code [Vlachas, 2019] with input dimension $d = 100$. Time is normalized by the Lyapunov exponent $\lambda = 0.043$ which defines the characteristic time of exponential divergence of a chaotic system, i.e. $|\delta x^{(t)}| \approx e^{\lambda t}|\delta x^{(0)}|$.

KS data points $\boldsymbol{i}^{(0)}, \dots, \boldsymbol{i}^{(t-1)}$ are fed to the algorithm. The output in Equation 3.2 for Reservoir Computing consists here in predicting the next state of the system : $\hat{\boldsymbol{o}}^{(t)} = \boldsymbol{i}^{(t)}$. This prediction is then used for updating the reservoir state in Equation 3.1, the algorithm outputs the next prediction $\hat{\boldsymbol{o}}^{(t+1)}$, and we repeat this operation. Thus, Reservoir Computing defines a trained *autonomous dynamical system* that one wants to be synchronized with the chaotic time series [Antonik, 2018].

The hyperparameters are found with a grid search, and the same set is used for RC, SRC, and RK to demonstrate their equivalence. To improve the performance of the final algorithm, we also add a random bias and use a concatenation of the reservoir state and the current input for prediction, replacing Equation 3.2 by $\hat{\boldsymbol{o}}^{t} = \mathbf{W_o}[\boldsymbol{x}^{(t)}, \boldsymbol{i}^{(t)}]$.

Prediction performance is presented in Figure 3.2. RC and SRC are trained on $n = 70{,}000$ training points and RK on a sub-sampling of 7,000 of these training points, due to memory constraints. The testing dataset length was set at 2,000. The sizes N in Reservoir Computing and Structured Reservoir Computing are chosen so the dimension $p = N + d$ in Equation 3.10 is a power of two for the multiplication by Hadamard matrix. Linear regression is solved using Cholesky decomposition.

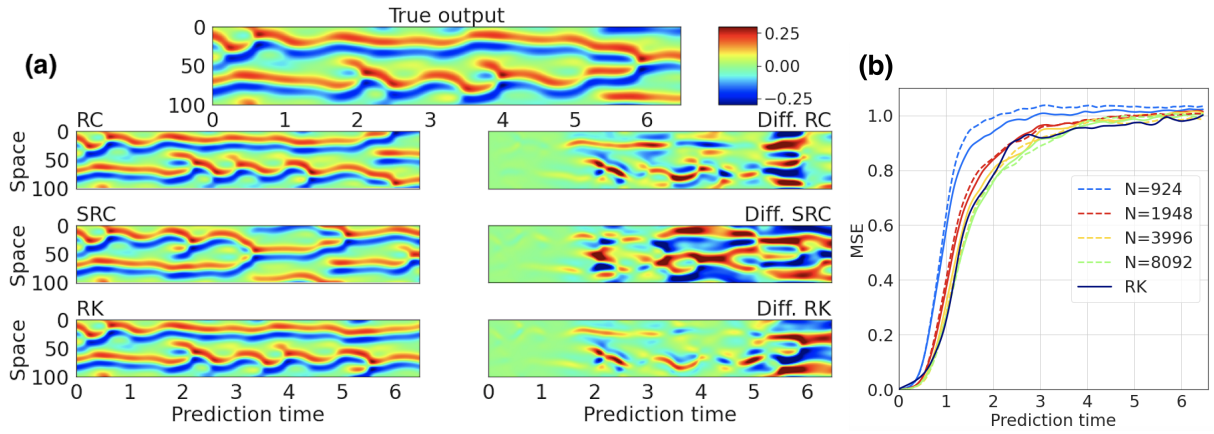


FIGURE 3.2 – (a) Comparison of different algorithms for the prediction of the Kuramoto-Sivashinsky dataset. True output (top), predictions of RC/SRC/RK (left) and differences with the true output (right), with reservoirs in RC/SRC of size $N = 3{,}996$. We observe that each technique is able to predict up to a few characteristic times. (b) Mean-Squared Error as a function of the prediction time for RC **(full lines)**, SRC **(dashed lines)**, and RK **(black)**. For all the reservoir sizes considered, the performances of RC and SRC are very close and they converge for large dimensions to the RK limit.

The predictions in Figure 3.2 show that all three algorithms are able to predict up to a few characteristic times. Since the prediction performance varies quite significantly between different realizations, we also display the Mean-Squared Error (MSE) of each algorithm, as a function of the prediction time and averaged over 10 realizations. We normalize each curve by the MSE between two independent KS systems.

We observe a decrease in the MSE when the size of the reservoir increases, meaning a larger reservoir yields better predictions. Performances are equivalent between RC and SRC, and they converge towards the RK performance for large reservoir sizes. Hence, this means RC, SRC, and RK can seamlessly replace one another in practical applications.

Timing benchmark. Several steps in the Reservoir Computing pipeline need to be assessed separately, as described in 3.3.3. We present the timings on a training set of length $n = 10,000$ and testing length of $m = 2,000$ in Table 3.2 for all three algorithms.

The *forward pass*, i.e. computing the recurrent iterations of each algorithm, is considered separately from the linear regression for *training*, to emphasize the cost of this important step. In RC, the most expensive operation is the dense matrix multiplication ; the GPU memory was not large enough to store the square weight matrix for the two largest reservoir sizes. With Structured Reservoir Computing, this forward pass becomes very efficient even at large sizes, and memory is not an issue anymore. We observe that the forward pass complexity becomes approximately constant until dimension $\sim 10^5$. On the other hand, Recurrent Kernels iterations are very fast, as we only need to compute element-wise operations in a kernel matrix.

Prediction requires a forward pass and then is performed with autonomous dynamics as presented on Figure 3.2 where Equation 3.2 is repeated 600 times. For Recurrent Kernels, prediction remains slow, and this drawback is exacerbated by the autonomous dynamics strategy in time series prediction, that requires successive prediction steps.

This shows that SRC is a very efficient way to scale-up Reservoir Computing to large sizes and reach the asymptotic limit of performance. On the other hand, the deterministic Recurrent Kernels are surprisingly fast to iterate, at the cost of a relatively slow prediction when the number of training samples n is large.

	$N = 1{,}948$	$N = 3{,}996$	$N = 8{,}092$	$N = 16{,}284$	$N = 32{,}668$
RC	**2.6**/0.02/1.9	**3.1**/0.05/4.6	10.4/0.16/15.4	Mem. Err.	Mem. Err.
SRC	3.3/0.02/**1.6**	3.4/0.05/**2.7**	**3.5**/0.16/**3.7**	**3.6**/0.57/**6.8**	**3.6**/2.57/**13.0**
RK	**0.7**/**0.09**/23.0				

TABLE 3.2 – Timing (Forward/Train/Predict, in seconds) for a KS prediction task as a function of N. We observe that Recurrent Kernels are surprisingly fast, except for prediction. Structured Reservoir Computing reduces drastically the speed of the forward pass at large sizes and is more memory-efficient than Reservoir Computing. Experiments were run on an NVIDIA V100 16GB.

3.5 Conclusion

In this work, we strengthened the connection between Reservoir Computing and kernel methods based on theoretical and numerical results, and showed how efficient implementations of Recurrent Kernels can be competitive with standard RC for chaotic time series prediction. Future lines of work include a deeper study of stability and the extension to different recurrent networks topologies. We deeply think this connection between random RNNs and kernel methods will open up future research on this important topic in machine learning.

We additionally introduced Structured Reservoir Computing, an acceleration technique of Reservoir Computing using fast Hadamard transforms. With only a simple change of the reservoir weights, we are able to speed up and reduce the memory cost of Reservoir Computing and therefore reach very large network sizes. We believe Structured Reservoir Computing offers a promising alternative to conventional Reservoir Computing, replacing it whenever large reservoir sizes are required.

3.6 Appendix

3.6.1 Convergence rate for translation-invariant kernels

Theorem 3.6.1. *(Rotation-invariant kernels) For the RI recurrent kernel defined in Equation 3.9, under the assumptions detailed above, and with $\Lambda = 2\sigma_r^2 L$ (note the factor 2 compared to Theorem 3.3.1). For all $t \in \mathbb{N}$, the following inequality is satisfied for any $\delta > 0$ with probability at least $1 - 2(t+1)\delta$:*

$$\left|\langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)} \rangle - k_{t+1}(\Delta^{(t)}, ..., \Delta^{(0)})\right| \leq \frac{1-\Lambda^{t+1}}{1-\Lambda}\Theta(N) \qquad \text{if} \quad \Lambda \neq 1 \tag{3.20}$$

$$\leq (t+1)\Theta(N) \qquad \text{if} \quad \Lambda = 1 \tag{3.21}$$

with $\Theta(N) = \frac{4\kappa^2 \log\frac{1}{\delta}}{3N} + 2\kappa^2\sqrt{\frac{2\log\frac{1}{\delta}}{N}}$.

Proof. Under the assumptions, Proposition 3.3.2 yields with probability greater than $1 - 2\delta$:

$$\left|\langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)} \rangle - k([\boldsymbol{x}^{(t)}, \boldsymbol{i}^{(t)}], [\boldsymbol{y}^{(t)}, \boldsymbol{j}^{(t)}])\right| \leq \frac{4\kappa^2 \log\frac{1}{\delta}}{3N} + 2\kappa^2\sqrt{\frac{2\log\frac{1}{\delta}}{N}} = \Theta(N). \tag{3.22}$$

It means the larger the reservoir, the more Random Features N we sample, and the more the inner product of reservoir states concentrates towards its expectation value, at a rate $O(1/\sqrt{N})$. We now apply this inequality recursively to complete the proof, based on the observation that both Equations 3.11 and 3.12 are equivalent to : $\left|\langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)} \rangle - k_{t+1}(\Delta^{(t)}, ..., \Delta^{(0)})\right| \leq (1 + \Lambda + \Lambda^2 + ... + \Lambda^t)\Theta(N)$.
For $t = 0$, provided $\boldsymbol{x}^{(0)} = \boldsymbol{y}^{(0)} = 0$, we have, according to Equation 3.14, with probability at least $1 - 2\delta$:

$$\left|\langle \boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)} \rangle - k_1(\Delta^{(0)})\right| \leq \Theta(N). \tag{3.23}$$

For any time $t \in \mathbb{N}^*$, let us assume the following event A_t is true with probability $\mathbb{P}(A_t) \geq 1 - 2t\delta$:

$$\left|\langle \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)} \rangle - k_t(\Delta^{(t-1)}, ..., \Delta^{(0)})\right| \leq (1 + ... + \Lambda^{t-1})\Theta(N). \tag{3.24}$$

Using the Lipschitz-continuity of k, this inequality is equivalent to :

$$\left|k(2\sigma_r^2(1 - \langle \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)} \rangle) + \Delta^{(t)}) - k(2\sigma_r^2(1 - k_t(\Delta^{(t-1)}, ..., \Delta^{(0)})) + \Delta^{(t)})\right| \leq (\Lambda + ... + \Lambda^t)\Theta(N). \tag{3.25}$$

With Equation 3.14, the following event B_t is true with probability $\mathbb{P}(B_t) \geq 1 - 2\delta$:

$$\left|\langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)} \rangle - k(2\sigma_r^2(1 - \langle \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)} \rangle) + \Delta^{(t)})\right| \leq \Theta(N). \tag{3.26}$$

Summing Equations 3.25 and 3.26, with the triangular inequality and a union bound, the following event A_{t+1} is true with probability $\mathbb{P}(A_{t+1}) \geq \mathbb{P}(B_t \cap A_t) = \mathbb{P}(B_t) + \mathbb{P}(A_t) - \mathbb{P}(B_t \cup A_t) \geq$

$1 - 2\delta + 1 - 2t\delta - 1 = 1 - 2(t+1)\delta$:

$$\left|\langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)}\rangle - k_{t+1}(\Delta^{(t)}, ..., \Delta^{(0)})\right| \leq (1 + \ldots + \Lambda^t)\Theta(N). \tag{3.27}$$

□

3.6.2 Explicit Recurrent Kernel formulas

We have defined so far the general formulas of RI and TI Recurrent Kernels in Equations 3.8 and 3.9. We will give now their explicit formulas for specific activation functions that one may encounter in Reservoir Computing.

Two reservoirs $\boldsymbol{x}^{(t)}$ and $\boldsymbol{y}^{(t)}$ are driven by two respective input time series $\boldsymbol{i}^{(t)}$ and $\boldsymbol{j}^{(t)}$. They obey Equation 3.1 and in the infinite-size limit, their inner product converges towards an explicit Recurrent Kernel. In practice, one needs to compute the inner products for each pair of input time series, from the training or testing sets, that we concatenate to construct a kernel matrix. A list of different activation functions and their associated kernels is provided in Table 3.3. Without recurrence, it is always possible to write the corresponding kernel as an integral that one may evaluate :

$$k(\boldsymbol{u}, \boldsymbol{v}) = \int d\boldsymbol{w}\rho(\boldsymbol{w})f(\langle \boldsymbol{w}, \boldsymbol{u}\rangle)f(\langle \boldsymbol{w}, \boldsymbol{v}\rangle); \tag{3.28}$$

where $\rho(\boldsymbol{w})$ is the distribution of the weights, usually an i.i.d. Gaussian distribution. However, in all the cases presented here, $k(\boldsymbol{u}, \boldsymbol{v})$ happens to contain inner products $\langle \boldsymbol{u}, \boldsymbol{v}\rangle$, which makes it possible to define the corresponding Recurrent Kernel.

In our case, $\boldsymbol{u}^{(t)} = [\sigma_r \boldsymbol{x}^{(t)}, \sigma_i \boldsymbol{i}^{(t)}]$ and $\boldsymbol{v}^{(t)} = [\sigma_r \boldsymbol{y}^{(t)}, \sigma_i \boldsymbol{j}^{(t)}]$ so that :

$$\langle \boldsymbol{u}^{(t)}, \boldsymbol{v}^{(t)}\rangle = \sigma_r^2 \langle \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}\rangle + \sigma_i^2 \langle \boldsymbol{i}^{(t)}, \boldsymbol{j}^{(t)}\rangle \to \sigma_r^2 k_t(l^{(t-1)}, \ldots, l^{(0)}) + l^{(t)} \tag{3.29}$$

when the reservoir size $N \to \infty$. Similarly, $\|\boldsymbol{u}^{(t)}\|^2 = \langle \boldsymbol{u}^{(t)}, \boldsymbol{u}^{(t)}\rangle$ and $\|\boldsymbol{v}^{(t)}\|^2$ are symmetric inner products that can similarly be expressed as in Equation 3.29. Hence, the Recurrent Kernel formulas are derived from the previous one by noting that :

$$\lim_{N\to\infty} \langle \boldsymbol{x}^{(t+1)}, \boldsymbol{y}^{(t+1)}\rangle = k_{t+1}(l^{(t)}, \ldots, l^{(0)}) \equiv k(\boldsymbol{u}^{(t)}, \boldsymbol{v}^{(t)}). \tag{3.30}$$

Analytic formulas in more general cases may not exist and they would need to be replaced by successive integrals. In this work, we restricted ourselves to functions described in Table 1 with simple analytic formulas, to speed up the RK computation. For instance, the error function is very close but not equal to the hyperbolic tangent in our implementations of Reservoir Computing, and performance in practice is very similar.

The successive integrals can still be explicitly defined. Equation 3.28 describes the asymptotic kernel limit for any arbitrary $(\boldsymbol{u}, \boldsymbol{v})$. To define recurrent kernels, we need to express it as a function of $\langle \boldsymbol{u}, \boldsymbol{v}\rangle$, $\|\boldsymbol{u}\|^2$, and $\|\boldsymbol{v}\|^2$ only. This is possible thanks to the invariance by rotation of the Gaussian distribution of w. Without loss of generality, we can thus assume that $\boldsymbol{u} = \|\boldsymbol{u}\|\boldsymbol{e}_1$ and $\boldsymbol{v} = \|\boldsymbol{v}\|(\cos\theta \boldsymbol{e}_1 + \sin\theta \boldsymbol{e}_2)$ with $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ the first two vectors of the canonical basis and

$f(\cdot)$	Associated kernel $k(u,v)$
Erf$(\cdot)$	$\frac{2}{\pi}\arcsin\left(\frac{2\langle \boldsymbol{u},\boldsymbol{v}\rangle}{\sqrt{(1+2\|\boldsymbol{u}\|^2)(1+2\|\boldsymbol{v}\|^2)}}\right)$
RFFs : $[\cos(\cdot),\sin(\cdot)]$	$\exp\left(-\frac{\|\boldsymbol{u}-\boldsymbol{v}\|^2}{2}\right) = \exp\left(-\frac{\|\boldsymbol{u}\|^2+\|\boldsymbol{v}\|^2-2\langle \boldsymbol{u},\boldsymbol{v}\rangle}{2}\right)$
Sign$(\cdot)$	$\frac{2}{\pi}\arcsin\left(\frac{\langle \boldsymbol{u},\boldsymbol{v}\rangle}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}\right)$
Heaviside$(\cdot)$	$\frac{1}{2}-\frac{1}{2\pi}\arccos\left(\frac{\langle \boldsymbol{u},\boldsymbol{v}\rangle}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}\right)$
ReLU$(\cdot)$	$\frac{1}{2\pi}\left(\langle \boldsymbol{u},\boldsymbol{v}\rangle \arccos(-\frac{\langle \boldsymbol{u},\boldsymbol{v}\rangle}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}) + \|\boldsymbol{u}\|\|\boldsymbol{v}\|\sqrt{1-\left(\frac{\langle \boldsymbol{u},\boldsymbol{v}\rangle}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}\right)^2}\right)$

TABLE 3.3 – Table of point-wise non-linearities f and their approximated kernels. For any $\boldsymbol{u},\boldsymbol{v}\in\mathbb{R}^p$ the kernel $k(\boldsymbol{u},\boldsymbol{v})$ is the limit when N goes to infinity of $\frac{1}{N}\langle f(\mathbf{W}\boldsymbol{u}), f(\mathbf{W}\boldsymbol{v})\rangle$ with $\mathbf{W}\in\mathbb{R}^{N\times p}$ an i.i.d. normal random matrix. In the case of Reservoir Computing, we have $\boldsymbol{u}=\boldsymbol{u}^{(t)}=[\sigma_r\boldsymbol{x}^{(t)},\sigma_i\boldsymbol{i}^{(t)}]$ and $\boldsymbol{v}=\boldsymbol{v}^{(t)}=[\sigma_r\boldsymbol{y}^{(t)},\sigma_i\boldsymbol{j}^{(t)}]$. We observe that in this table, all kernel formulas depend only on $\langle \boldsymbol{u},\boldsymbol{v}\rangle$, $\|\boldsymbol{u}\|$, and $\|\boldsymbol{v}\|$, which makes it possible to easily derive the Recurrent Kernel equations.

$\theta = \langle \boldsymbol{u},\boldsymbol{v}\rangle/(\|\boldsymbol{u}\|\|\boldsymbol{v}\|)$ (which is a function of the three quantities of interest). The multidimensional integral boils down to a two dimensional integral :

$$k(\boldsymbol{u},\boldsymbol{v}) = \int\int d\boldsymbol{w}_1 d\boldsymbol{w}_2 \rho(\boldsymbol{w}_1)\rho(\boldsymbol{w}_2) f(\boldsymbol{w}_1\|\boldsymbol{u}\|) f(\|\boldsymbol{v}\|(\boldsymbol{w}_1\cos\theta + \boldsymbol{w}_2\sin\theta)), \tag{3.31}$$

where $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are Gaussian random variables, projections of $\boldsymbol{w}$ on $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. Hence it is possible to iterate Recurrent Kernels numerically, that are the large-size limit of any Reservoir Computing algorithm for every activation function f. Each component of the square kernel matrix would require the evaluation of this two-dimensional integral, it may be possible to use tabular values to speed up computation.

3.6.3 Numerical study of the independence hypothesis

One assumption for the previous convergence theorems states the weight matrices $\mathbf{W_r}$ and $\mathbf{W_i}$ have to be redrawn at each iteration. This independence hypothesis is required in Equations 3.18 and 3.26, to ensure that $\boldsymbol{x}^{(t)}$ and $\boldsymbol{y}^{(t)}$ are uncorrelated with the weight matrices. This is necessary in the theoretical study to properly define the expectations and ensure the i.i.d. requirement for the random variables in the Bernstein inequality.
However, this assumption is unrealistic for practical Reservoir Computing. Resampling weight matrices at each timestep is computationally demanding and output weights would depend on the realization of these random matrices : one would need to keep the same random matrices in memory for testing.
However, in Figure 3.3, we investigate the convergence with and without redrawing weights at each iteration, and this independence hypothesis does not seem to be necessary : convergence is still achieved with fixed weight matrices. We show the Mean-Squared Error $\|\boldsymbol{K}_1-\boldsymbol{K}_2\|_2^2/n^2$

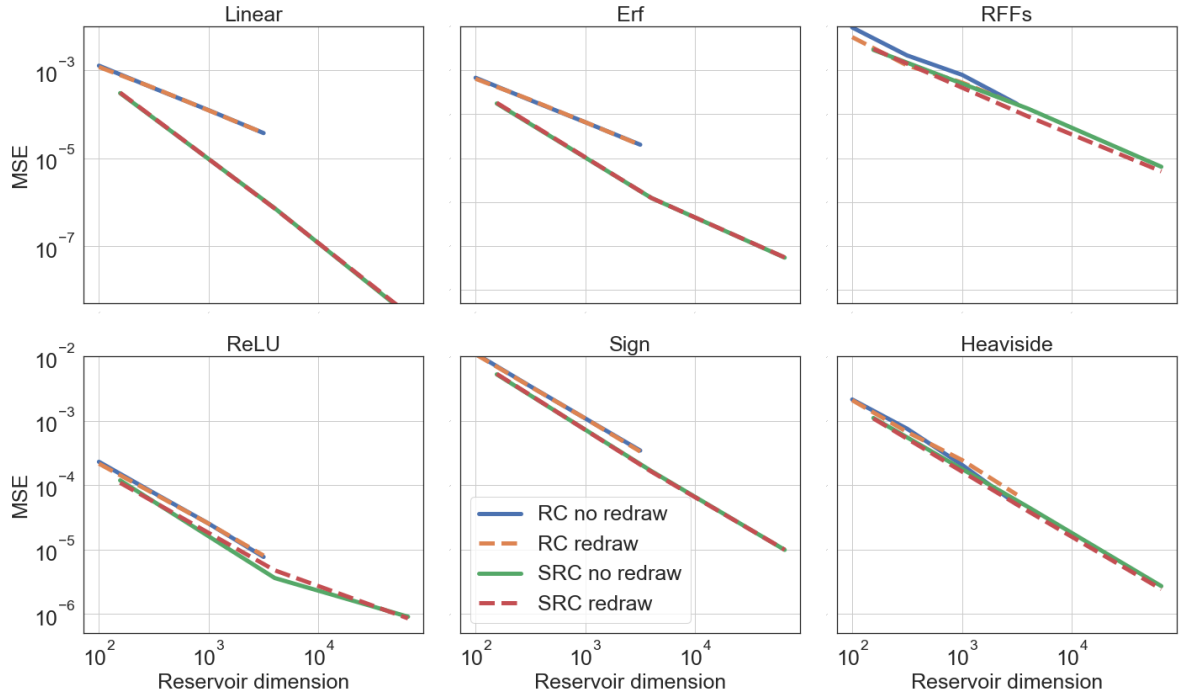


FIGURE 3.3 – Mean-Squared error between the kernel matrix obtained with RC/SRC with the asymptotic kernel limit, with and without resampling the random matrices at each iteration, to test the independence hypothesis of the theorem. 50×50 kernel matrices have been generated for all pairs of 50 random input time series of length 10. Several activation functions and their corresponding recurrent kernels are presented here. We observe that the hypothesis does not seem to be necessary since RC and SRC without resampling also converge to the RK limit at sensibly the same speed.

between the kernel matrix $\boldsymbol{K}_1$ from the explicit RK formula and $\hat{\boldsymbol{K}}_2$ the one obtained with RC and SRC, with and without redrawing the random matrices at every timestep. Each kernel matrix is of size 50×50, as we use $n = 50$ random i.i.d Gaussian input time series of dimension 50 and time length 10. Each curve is an average over 10 realizations and the reservoir scale is set to $\sigma_r^2 = 0.25$ to ensure stability.

We confirm the observation from Figure 3.1 that the larger the reservoir dimension, the closer we are from the RK asymptotic limit. This is valid for several activation functions, the ones presented in Table 3.3. We also confirm that SRC generally converges faster than RC.

Convergence is still achieved when resampling the weights at each iteration, and speed of convergence is not significantly different than for the fixed random matrix case. Thus convergence seems to be much more robust in practice, and this may call for further theoretical studies.

3.6.4 Stability of Reservoir Computing and Recurrent Kernels

As the reservoir is itself a dynamical system, it can be stable (differences in initial conditions vanish with time) or chaotic (differences in initial conditions explode exponentially). This is linked with the Echo-State Property, extensively studied for Reservoir Computing. It states that two reservoirs initialized differently need to converge to the same trajectory, provided they share the

same weights (at each time step if weights are resampled). This property is important so that the reservoir state after a large enough time τ does not depend on the arbitrary reservoir initialization. Stability or chaos can be tuned depending on a set of hyperparameters. An important one is the scale of the reservoir weights : when small, initial differences get damped exponentially with time, whereas they may explode if reservoir weights are large.

We verify this Echo-State Property here for Reservoir Computing. In Figure 3.4 we present the squared distance $\|\boldsymbol{x}_1^{(t)} - \boldsymbol{x}_2^{(t)}\|^2$ as a function of time t between two randomly initialized reservoirs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, for the same input time series from the Kuramoto-Sivashinsky dataset. A normalization factor has been added to normalize this distance to 1 at $t = 0$ and each curve is an average over 100 realizations. The activation is the error function, the input scale is set to a small value $\sigma_i^2 = 0.01$, and we vary the reservoir scale σ_r^2. For $\sigma_r^2 = 0.49$ and 1, dynamics are stable and the two reservoir states converge quite quickly to the same trajectory. When $\sigma_r^2 = 2.25$, dynamics becomes chaotic and the two reservoirs follow very different dynamics due to their different initial conditions.

Recurrent Kernels may also present this transition from stability to chaos. Moreover, this stability property is important for Recurrent Kernels in practice. RKs need to be iterated a certain number of times, and thanks to stability this number of iterations can be reduced to the finite memory τ and not on the full length of the time series. This change reduces considerably the computational costs.

We thus also investigate numerically the stability of Recurrent Kernels, i.e. how they depend on the initial conditions. In Figure 3.4, we present the normalized difference between two kernel matrices $\|\boldsymbol{K}_1^{(t)} - \boldsymbol{K}_2^{(t)}\|_2^2$ as a function of time, for two recurrent kernels $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ initialized with a matrix full of ones or of zeros, and fed with the same input time series, for the arcsine Recurrent Kernel corresponding to the erf activation function. We observe that Recurrent Kernels are in general a lot more stable than Reservoir Computing. This characteristic may be interesting to investigate further.

We may now draw an interesting parallel between this study and, as we unroll the Recurrent Neural Network through time, multilayer perceptrons with random weights, linked with compositional kernels. They correspond to our case, $\boldsymbol{i}^{(t)} = 0$ for $t \geq 1$ and $\boldsymbol{i}^{(0)} \in \mathbb{R}^d$ is the time-independent input. This stability property corresponds to a final layer that does not depend on $\boldsymbol{i}^{(0)}$, and as such information does not flow in the deep network. Hence, whereas it is advantageous in Reservoir Computing to be stable, it may be detrimental for deep neural networks.

3.6.5 Implementation details for Reservoir Computing

Several tweaks are useful to improve the performance of Reservoir Computing for time series prediction. We used the erf activation function as it is the closest from the hyperbolic tangent already used in Reservoir Computing, that still possess a simple Recurrent Kernel formula.

First, we add a random additive bias $\boldsymbol{b} \in \mathbb{R}^N$ sampled from an i.i.d. normal distribution $\mathcal{N}(0, \sigma_b^2)$. The variance of this bias vector σ_b^2 is a hyperparameter to tune, like the variance of the reservoir or input weights. This bias helps to diversify the neuron activations in the reservoir. Hence, the

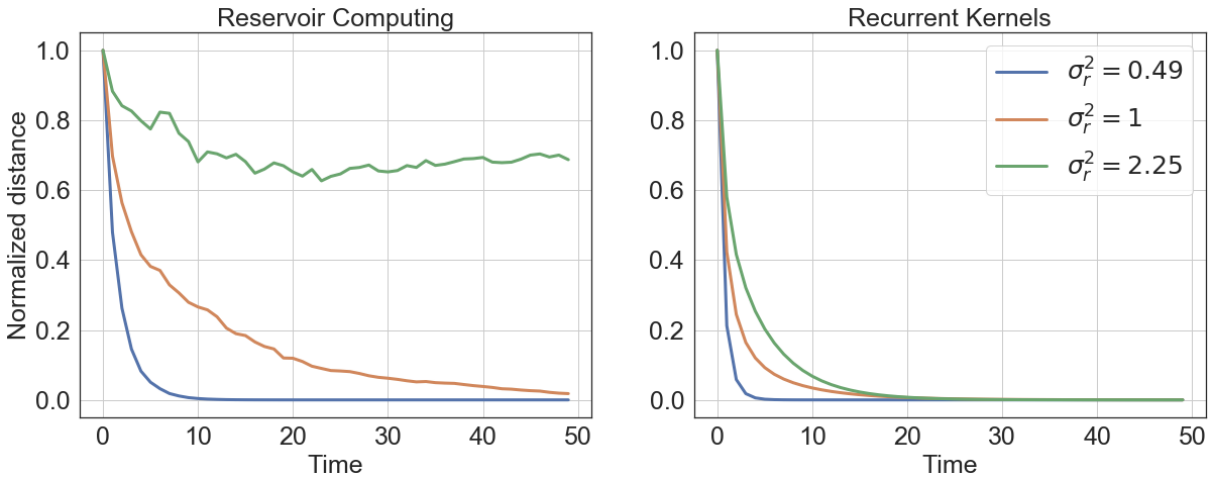


FIGURE 3.4 – Stability analysis of Reservoir Computing and Recurrent Kernels. We compute the normalized square distance between two reservoirs or recurrent kernels fed with the same input time series and different initializations. For RK or RC when $\sigma_r^2 \leq 1$ we see that trajectories converge to a single one after some time. This means that initial conditions are forgotten after a number of iterations. On the other hand, when $\sigma_r^2 = 2.25$ for Reservoir Computing, the reservoir is in a chaotic regime and always depend on initial conditions. It is interesting to observe that Recurrent Kernels are generally more stable than RC.

reservoir update equation becomes :

$$\boldsymbol{x}^{(t+1)} = \frac{1}{\sqrt{N}} f\left(\mathbf{W_r}\,\boldsymbol{x}^{(t)} + \mathbf{W_i}\,\boldsymbol{i}^{(t)} + \boldsymbol{b}\right). \tag{3.32}$$

As stated previously, we concatenate the reservoir state with the last value of the time series we have received. Information about the past is still encoded in the reservoir, but with this simple change, the reservoir is rather used to compute perturbations on the current value, and does not have to reconstruct the whole spatial profile. We add a renormalization hyperparameter r for this concatenation, in order to control the weight of the reservoir versus current input.
A hyperparameter search was performed, for a total of 5 hyperparameters (the reservoir scale, input scale, bias scale, the previous concatenation factor, regularization constant). Since there is a large number of hyperparameters to tune, we perform it on one hyperparameter at a time, going through the set of parameters several times. The final set of hyperparameters of Figure 3.2 is $\{\sigma_i, \sigma_r, \sigma_b, r, \alpha\} = \{0.4, 0.9, 0.4, 1.1, 10^{-2}\}$.
For completeness, we give here the exact definition of the Mean-Squared Error of Figure 3.2. The target output $\boldsymbol{O}(t) \in \mathbb{R}^d$ for $t = 1, \ldots, T_{\text{pred}}$ corresponds to the next states of the chaotic systems, and for each t, we evaluate the MSE between $\boldsymbol{O}(t)$ and the prediction of the algorithm $\hat{\boldsymbol{o}}(t)$, which is simply $\|\boldsymbol{O}(t) - \hat{\boldsymbol{o}}(t)\|^2/d$.

3.6.6 Implementation details for Recurrent Kernels

We also used a Recurrent Kernel to perform chaotic time series prediction. We chose an arcsine rotation-invariant kernel, the asymptotic limit of a reservoir with error function activations. We use the principle described in Section 3.6.2, with the addition of a random Gaussian bias that

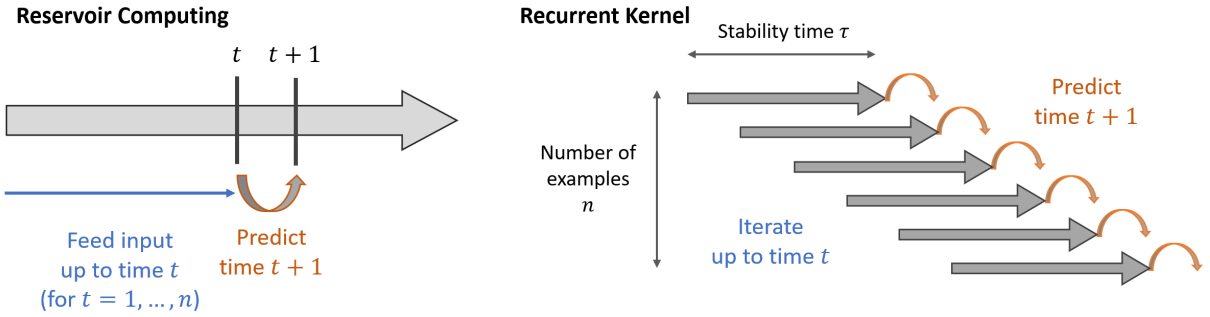


FIGURE 3.5 – How to use Recurrent Kernels for time series prediction. In Reservoir Computing, the input is continuously fed to the reservoir and all the reservoir states for every timestep t are stored for training. With Recurrent Kernels, we construct n small windows of the time series of length τ and compute scalar products between each pair using τ iteration of Equation 3.8 or 3.9.

corresponds to adding a constant dimension to the vector $\boldsymbol{u}^{(t)} = [\sigma_r \boldsymbol{x}^{(t)}, \sigma_i \boldsymbol{i}^{(t)}, \sigma_b]$.
Additionally, we have introduced for Reservoir Computing a concatenation step we need to reproduce with Recurrent Kernels. In RC, we concatenate the reservoir and the current input before computing the prediction. The corresponding operation for Recurrent Kernels is the addition of a linear kernel computed from all pairs of input points : $\boldsymbol{K}^+_{kl} = \langle \boldsymbol{i}_k^{(t)}, \boldsymbol{i}_l^{(t)} \rangle$. This kernel matrix $\boldsymbol{K}^+$ is added to the Recurrent Kernel after the iterations and before the linear model for prediction.
We also expand more on the process of generating the input data for Recurrent Kernels. In time series prediction, each reservoir state (neglecting a warm-up phase) is used during training to learn output weights to predict the future states of the system. Since there are n training examples, this corresponds to an $n \times n$ kernel matrix. In the Recurrent Kernel setting, we train a linear model on the final kernel matrix. We thus construct n time series of length $\tau = 50$ for each time step of the training data (neglecting the effect of edges), where the length τ is determined by the stability of the Recurrent Kernel. This process is depicted in Figure 3.5.

3.6.7 Recursive vs non-recursive prediction

Following previous strategies developed for chaotic time series prediction with Reservoir, RC, SRC, and RK algorithms were trained only to perform next-time-step prediction. To predict further in the future, this prediction is then fed back into the algorithm to iterate further in time. As explained previously, this defines an autonomous dynamical system that should be synchronized with the chaotic time series if training is successful.
Another possible strategy would be to use a given reservoir state to predict T_{pred} time steps in the future. The output dimension $c = d\,T_{\text{pred}}$ is larger and the learning task becomes more difficult.
We show here the usefulness of this strategy based on autonomous dynamics. In Figure 3.6, we show the performance of Reservoir Computing prediction on the Kuramoto-Sivashinsky dataset, with and without recursive prediction. With recursive prediction (left), this corresponds to the strategy already presented in Figure 3.2, and it is not surprising that prediction up to at least

2 Lyapunov exponents is possible. Without recursive prediction (right), the algorithm has a much harder time to predict the future of the chaotic system. Instead, after a short while, it only returns the average value of the time series.
Note that the same hyperparameters were used in both cases. While it may be possible to improve the performance of the direct prediction strategy, by increasing the size of the reservoir or playing with regularization parameter, but we show here the simplicity and effectiveness of the recursive prediction strategy.

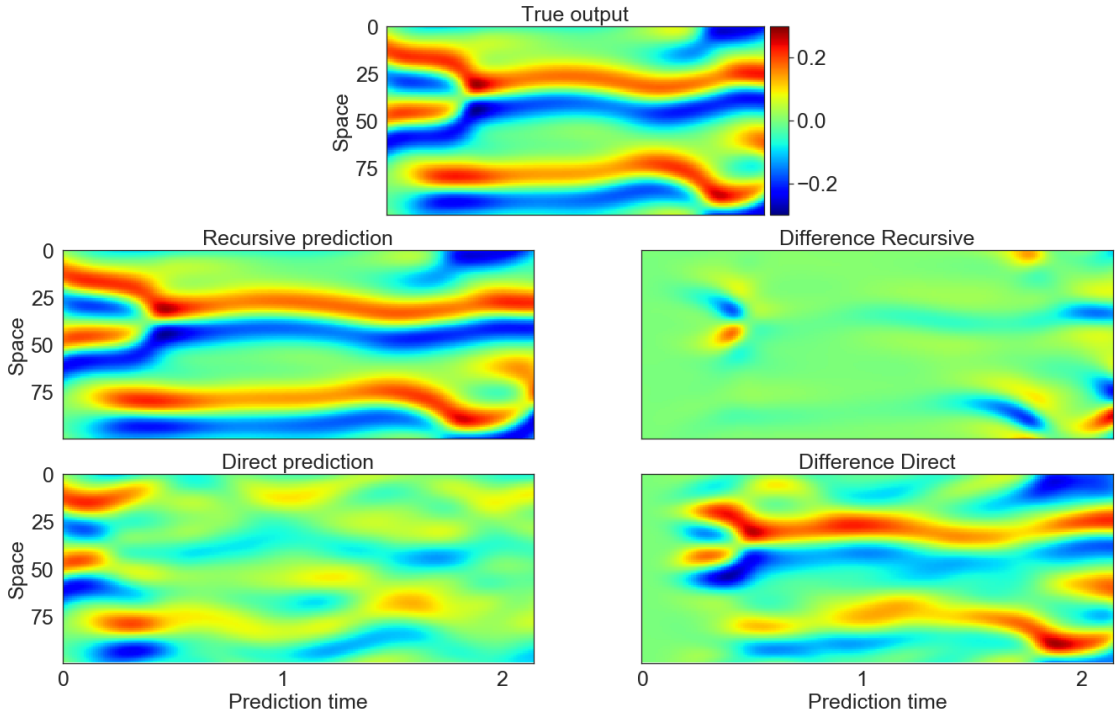


FIGURE 3.6 – Comparison of recursive and non-recursive prediction. We see that with recursive prediction (left), Reservoir Computing is able to predict quite precisely up to at least 2 characteristic times. On the other hand, without recursive prediction, Reservoir Computing quickly has a hard time to guess the future of the KS system and outputs its mean for long prediction times.

Chapter 4

Optical Random Features and their Kernel Limit

This chapter is based on [Ohana, 2020].

Approximating kernel functions with random features (RFs) has been a successful application of random projections for nonparametric estimation. However, performing random projections presents computational challenges for large-scale problems. Recently, a new optical hardware called Optical Processing Unit (OPU) has been developed for fast and energy-efficient computation of large-scale RFs in the analog domain. More specifically, the OPU performs the multiplication of input vectors by a large random matrix with complex-valued i.i.d. Gaussian entries, followed by the application of an element-wise squared absolute value operation – this last nonlinearity being intrinsic to the sensing process. In this Chapter, we show that this operation results in a dot-product kernel that has connections to the polynomial kernel, and we extend this computation to arbitrary powers of the feature map. Experiments demonstrate that the OPU kernel and its RF approximation achieve competitive performance in applications using kernel ridge regression and transfer learning for image classification. Crucially, thanks to the use of the OPU, these results are obtained with time and energy savings.

4.1 Introduction

Kernel methods represent a successful class of Machine Learning models, achieving state-of-the-art performance on a variety of tasks with theoretical guarantees [Schölkopf, 2002 ; Rudi, 2017a ; Caponnetto, 2007]. Applying kernel methods to large-scale problems, however, poses computational challenges, and this has motivated a variety of contributions to develop them at scale ; see, e.g., [Rudi, 2017a ; Smola, 2000 ; Zhang, 2013 ; Rudi, 2015 ; Cutajar, 2017].
Consider a supervised learning task, and let $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ be a set of n inputs with $\boldsymbol{x}_i \in \mathbb{R}^d$ associated with a set of labels $\{t_1, \dots, t_n\}$. In kernel methods, it is possible to establish a mapping

between inputs and labels by first mapping the inputs to a high-dimensional (possibly infinite dimensional) Hilbert space $\mathcal{H}$ using a nonlinear feature map $\boldsymbol{\varphi} : \mathbb{R}^d \to \mathcal{H}$, and then to apply the model to the transformed data. What characterizes these methods is that the mapping $\boldsymbol{\varphi}(\cdot)$ does not need to be specified and can be implicitly defined by choosing a kernel function $k(\cdot, \cdot)$. While kernel methods offer a flexible class of models, they do not scale well with the number n of data points in the training set, as one needs to store and perform algebraic operations with the kernel matrix $\mathbf{K}$, whose entries are $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and which require $\mathcal{O}(n^2)$ storage and $\mathcal{O}(n^3)$ operations.

In a series of celebrated papers [Rahimi, 2008 ; Rahimi, 2009], Rahimi and Recht have proposed approximation techniques of the kernel function using random features (RFs), which are based on random projections of the original features followed by the application of a nonlinear transformation. In practice, the kernel function is approximated by means of the scalar product between finite-dimensional random maps $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^D$:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\varphi}(\boldsymbol{x}_i), \boldsymbol{\varphi}(\boldsymbol{x}_j) \rangle_{\mathcal{H}} \approx \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_j). \tag{4.1}$$

The RF-based approximation turns a kernel-based model into a linear model with a new set of nonlinear features $\boldsymbol{\phi}(\boldsymbol{x})$; as a result, the computational complexity is reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(ndD)$ to construct the random features and $\mathcal{O}(D^3)$ to optimize the linear model, where D is the RF dimension and n the number of data points. Furthermore, there is no need to allocate the kernel matrix, reducing the storage from $\mathcal{O}(n^2)$ to $\mathcal{O}(nD) + \mathcal{O}(D^2)$. Unless approximation strategies to compute random features are used, e.g., [Le, 2013], computing RFs is one of the main computational bottlenecks. More details about kernel methods and random features can be found in Section 2.2.

A completely different approach was pioneered in Saade *et al.* [Saade, 2016a], where the random projections are instead made via an analog optical device – the Optical Processing Unit (OPU) – that performs these random projections literally at the speed of light and without having to store the random matrix in memory. Their results demonstrate that the OPU makes a significant contribution towards making kernel methods more practical for large-scale applications with the potential to drastically decrease computation time and memory, as well as power consumption. The OPU has also been applied to other frameworks like reservoir computing [Dong, 2018 ; Dong, 2019] and anomaly detection [Keriven, 2018].

Building on the milestone work of [Saade, 2016a], the goal of the present contribution is threefold : a) we derive in full generality the kernel to which the dot product computed by the OPU RFs converges, generalizing the earlier computation of [Saade, 2016a] to a larger class of kernels ; b) we present new examples and a benchmark of applications for the kernel of the OPU ; and c) we give a detailed comparison of the running time and energy consumption between the OPU and a last generation GPU.

4.2 The Optical Processing Unit

The principle of the random projections performed by the Optical Processing Unit (OPU) is based on the use of a heterogeneous material that scatters the light that goes through it, see Figure 4.1 for the experimental setup. The data vector $\boldsymbol{x} \in \mathbb{R}^d$ is encoded into light using a digital micromirror device (DMD). This encoded light then passes through the heterogeneous medium, performing the random matrix multiplication. As discussed in [Liutkus, 2014], light going through the scattering medium follows many extremely complex paths, that depend on refractive index inhomogeneities at random positions. For a fixed scattering medium, the resulting process is still linear, deterministic, and reproducible. Reproducibility is important as all our data vectors need to be multiplied by the same realisation of the random matrix.
After going through the "random" medium, we observe a speckle figure on the camera, where the light intensity at each point is modelled by a sum of the components of $\boldsymbol{x}$ weighted by random coefficients. Measuring the intensity of light induces a non-linear transformation of this sum, leading to :

Proposition 4.2.1. *Given a data vector $\boldsymbol{x} \in \mathbb{R}^d$, the random feature map performed by the Optical Processing Unit is :*

$$\boldsymbol{\phi}(\boldsymbol{x}) = \frac{1}{\sqrt{D}}|\mathbf{U}\boldsymbol{x}|^2, \tag{4.2}$$

where $\mathbf{U} \in \mathbb{C}^{D\times d}$ is a complex Gaussian random matrix whose elements $U_{i,j} \sim \mathcal{CN}(0,1)$, the variance being set to one without loss of generality, and depends on a multiplicative factor combining laser power and attenuation of the optical system. We will name these RFs optical random features.

More details about the OPU are given in Section 2.4.

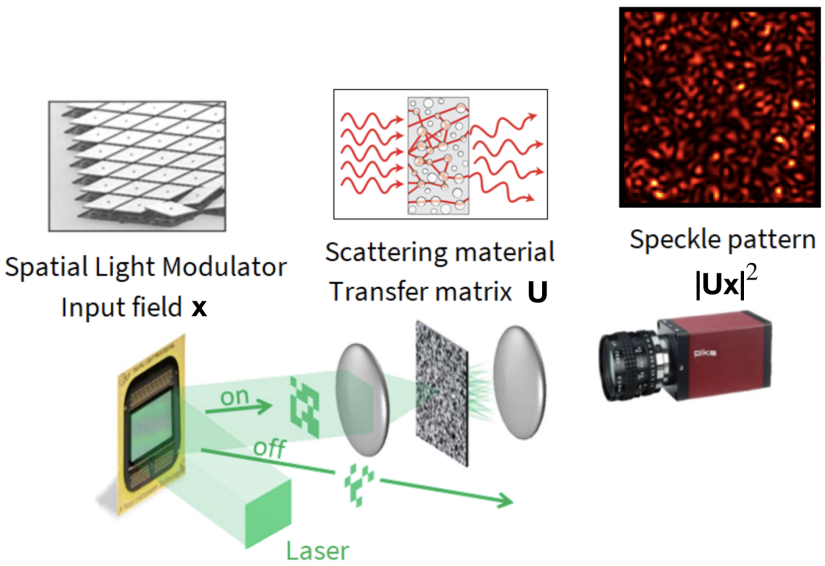


FIGURE 4.1 – Experimental setup of the Optical Processing Unit (modified with permission from [Saade, 2016a]). The data vector is encoded in the coherent light from a laser using a DMD. Light then goes through a scattering medium and a speckle pattern is measured by a camera.

4.3 Computing the kernel

When we map two data points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ into a feature space of dimension D using the optical RFs of Equation 4.2, we have to compute the following to obtain the associated kernel k_2 :

$$k_2(\boldsymbol{x}, \boldsymbol{y}) \approx \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{y}) = \frac{1}{D} \sum_{j=1}^{D} |\boldsymbol{x}^\top \boldsymbol{u}^{(j)}|^2 |\boldsymbol{y}^\top \boldsymbol{u}^{(j)}|^2 \tag{4.3}$$

$$\overset{D \to +\infty}{=} \int |\boldsymbol{x}^\top \boldsymbol{u}|^2 |\boldsymbol{y}^\top \boldsymbol{u}|^2 \mu(\boldsymbol{u}) \mathrm{d}\boldsymbol{u}, \tag{4.4}$$

with $\boldsymbol{u} = [\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(D)}]^\top$ and $\boldsymbol{u}^{(j)} \in \mathbb{R}^d, \forall j \in \{1, .., D\}$.

Theorem 4.3.1. *The kernel k_2 approximated by the dot product of optical random features of Equation 4.2 is given by :*

$$k_2(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2 + (\boldsymbol{x}^\top \boldsymbol{y})^2, \tag{4.5}$$

where the norm is the l_2 norm.

Proof. By rotational invariance of the complex Gaussian distribution, we can fix $\boldsymbol{x} = \|\boldsymbol{x}\| \mathbf{e_1}$ and $\boldsymbol{y} = \|\boldsymbol{y}\| (\mathbf{e_1} \cos\theta + \mathbf{e_2} \sin\theta)$, with θ being the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$, $\mathbf{e_1}$ and $\mathbf{e_2}$ being two orthonormal vectors. Letting $\mathbf{e_i}^\top \boldsymbol{u} = u_i \sim \mathcal{CN}(0, 1)$, $i = 1, 2$ and u_1^* be the complex conjugate of u_1, we obtain :

$$\begin{aligned} k_2(\boldsymbol{x}, \boldsymbol{y}) &= \|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2 \int |u_1|^2 |u_1 \cos\theta + u_2 \sin\theta|^2 \mathrm{d}\mu(\boldsymbol{u}) \\ &= \|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2 \int \Big(|u_1|^4 \cos^2\theta + |u_1|^2 |u_2|^2 \sin^2\theta + 2|u_1|^2 \mathrm{Re}(u_1^* u_2) \cos\theta \sin\theta \Big) \mathrm{d}\mu(u_1) \mathrm{d}\mu(u_2). \end{aligned}$$

By a parity argument, the third term in the integral vanishes, and the remaining ones can be explicitly computed, yielding :

$$k_2(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2 (1 + \cos^2\theta) = \|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2 + (\boldsymbol{x}^\top \boldsymbol{y})^2.$$

Two extended versions of the proof are presented in Appendix 4.5.1. □

Numerically, one can change the exponent of the feature map to $m \in \mathbb{R}^+$, which, using notations of Equation 4.2, becomes :

$$\boldsymbol{\phi}(\boldsymbol{x}) = \frac{1}{\sqrt{D}} |\mathbf{U}\boldsymbol{x}|^m. \tag{4.6}$$

Theorem 4.3.2. *When the exponent m is even, i.e. $m = 2s$, $\forall s \in \mathbb{N}$, the dot product of feature maps of Equation 4.6 tends to the kernel k_{2s} (for $D \to \infty$) :*

$$k_{2s}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x}\|^m \|\boldsymbol{y}\|^m \sum_{i=0}^{s} (s!)^2 \binom{s}{i}^2 \frac{(\boldsymbol{x}^\top \boldsymbol{y})^{2i}}{\|\boldsymbol{x}\|^{2i} \|\boldsymbol{y}\|^{2i}}. \tag{4.7}$$

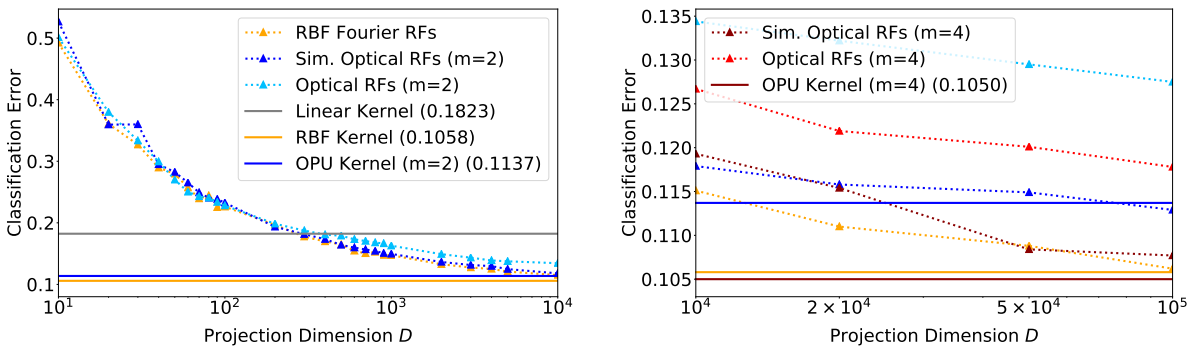
The proof is given in Appendix 4.5.3. Moreover, a generalization $\forall m \in \mathbb{R}^+$ can be established.

Equation 4.7 is connected to the polynomial kernel [Schölkopf, 2002] defined as :

$$(\nu + \boldsymbol{x}^\top \boldsymbol{y})^p = \sum_{i=0}^{p} \binom{p}{i} \nu^{p-i} (\boldsymbol{x}^\top \boldsymbol{y})^i, \tag{4.8}$$

with $\nu \geq 0$ and $p \in \mathbb{N}$ the order of the kernel. For $\nu = 0$ the kernel is called homogeneous. For $\nu > 0$ the polynomial kernel consists of a sum of lower order homogeneous polynomial kernels up to order p. It can be seen as having richer feature vectors including all lower-order kernel features. For optical RFs raised to the power of $s \in \mathbb{N}$ we have a sum of homogeneous polynomial kernels taken to even orders up to $m = 2s$.
Since $\boldsymbol{x}^\top \boldsymbol{y} = \|\boldsymbol{x}\| \|\boldsymbol{y}\| \cos\theta$, the kernel scales with $\|\boldsymbol{x}\|^i \|\boldsymbol{y}\|^i$, which is characteristic to any homogeneous polynomial kernel. It is easy to extend this relation to the inhomogeneous polynomial kernel by appending a bias to the input vectors, i.e. $\boldsymbol{x'}^\top \boldsymbol{y'} = \nu + \boldsymbol{x}^\top \boldsymbol{y}$ when $\boldsymbol{x'} = (\sqrt{\nu}, \boldsymbol{x}_1, ..., \boldsymbol{x}_d)^\top$ and $\boldsymbol{y'} = (\sqrt{\nu}, \boldsymbol{y}_1, ..., \boldsymbol{y}_d)^\top$. A practical drawback of this approach is that increasing the power of the optical RFs also increases their variance. Thus, convergence requires higher projection dimensions. Although high dimensional projections can be computed easily using the OPU, solving models on top of them poses other challenges that require special treatment [Rudi, 2017a] (e.g. Ridge Regression scales cubically with D). Therefore, we did not include these cases in the experiments in the next section and leave them for future research.

4.4 Experiments



(a) $D \leq 10^4$ (without optical RFs for $m = 4$)

(b) $D \geq 10^4$ (with optical RFs for $m = 4$)

FIGURE 4.2 – Ridge Regression test error on Fashion-MNIST for different RFs and projection dimensions D. Horizontal lines show the test error using the true kernel. Standard deviations for different seeds are negligibly small and not shown in the plot. Plot (a) compares optical RFs of degree $m = 2$ to RBF Fourier RFs. Higher degree optical RFs are left out for better readability. The more slowly converging optical RFs for $m = 4$ are added for larger D in plot (b).

In this section, we assess the usefulness of optical RFs for different settings and datasets. The model of our choice in each case is Ridge Regression. OPU experiments were performed remotely on the OPU prototype "Vulcain", running in the LightOn Cloud with library `LightOnOPU` v1.0.2. Since the current version only supports binary input data we decide to binarize inputs for all

experiments using a threshold binarizer (see Appendix 4.5.5). The code of the experiments is publicly available [1].

4.4.1 Optical random features for Fashion-MNIST

We compare optical RFs (simulated as well as physical) to an RBF Fourier Features baseline for different projection dimensions D on Fashion-MNIST. We use individually optimized hyperparameters for all RFs that are found for $D = 10^4$ using an extensive grid search on a held-out validation set. The same hyperparameters are also used for the precise kernel limit. Figure 4.2 shows how the overall classification error decreases as D increases. Part (b) shows that simulated optical RFs for $m = 2$ and RBF Fourier RFs reach the respective kernel test score at $D = 10^5$. Simulated optical RFs for $m = 4$ converge more slowly but outperform $m = 2$ features from $D = 2 \times 10^4$. They perform similarly well as RBF Fourier RFs at $D = 10^5$. The performance gap between $m = 2$ and $m = 4$ also increases for the real optical RFs with increasing D. This gap is larger than for the simulated optical RFs due to an increase in regularization for the $m = 2$ features that was needed to add numerical stability when solving linear systems for large D. The real OPU loses around 1.5% accuracy for $m = 2$ and 1.0% for $m = 4$ for $D = 10^5$, which is due slightly suboptimal hyperparameters to improve numerical stability for large dimensions. Moreover, there is a small additional loss due to the quantization of the analog signal when the OPU camera records the visual projection.

4.4.2 Transfer learning on CIFAR-10

Architecture	ResNet34				AlexNet			VGG16		
Layer	L1	L2	L3	Final	MP1	MP2	Final	MP4	MP5	Final
Dimension d	4 096	2 048	1 024	512	576	192	9 216	2 048	512	25 088
Sim. Opt. RFs	30.4	**24.7**	**28.9**	**11.6**	**38.1**	**41.9**	19.6	**20.7**	**29.8**	15.2 (**12.9**)
Optical RFs	31.1	25.7	29.7	12.3	39.2	42.6	20.8	21.5	30.2	16.4
RBF Four. RFs	**30.1**	25.2	30.0	12.3	39.4	**41.9**	**19.1**	**20.7**	30.1	14.8 (13.0)
No RFs	31.3	26.7	33.5	14.7	44.6	48.8	19.6	22.5	34.8	**13.3**

TABLE 4.1 – Test errors (in %) on CIFAR-10 using $D = 10^4$ RFs for each kernel (except linear). Features were extracted from intermediate layers when using the original input size (32x32). Final convolutional layers were used with upscaled inputs (224x224). L(i) refers to the ith ResNet34 layer and MP(i) to the ith MaxPool layer of VGG16/AlexNet. Values for the kernel limit are shown in parenthesis (last column).

An interesting use case for the OPU is transfer learning for image classification. For this purpose we extract a diverse set of features from the CIFAR-10 image classification dataset using three different convolutional neural networks (ResNet34 [He, 2015], AlexNet [Krizhevsky, 2012] and VGG16 [Simonyan, 2014a]). The networks were pretrained on the well-known ImageNet classification benchmark [Russakovsky, 2015]. For transfer learning, we can either fine-tune these networks and therefore the convolutional features to the data at hand, or we can directly apply a

1. https://github.com/joneswack/opu-kernel-experiments

classifier on them assuming that they generalize well enough to the data. The latter case requires much less computational resources while still producing considerable performance gains over the use of the original features. This light-weight approach can be carried out on a CPU in a short amount of time where the classification error can be improved with RFs.

We compare Optical RFs and RBF Fourier RFs to a simple baseline that directly works with the provided convolutional features (no RFs). Table 4.1 shows the test errors achieved on CIFAR-10. Each column corresponds to convolutional features extracted from a specific layer of one of the three networks.

Since the projection dimension $D = 10^4$ was left constant throughout the experiments, it can be observed that RFs perform particularly well compared to a linear kernel when $D \gg d$ where d is the input dimension. For the opposite case $D \ll d$ the lower dimensional projection leads to an increasing test error. This effect can be observed in particular in the last column where the test error of the RF approximation is higher than without RFs. The contrary can be achieved with large enough D as indicated by the values for the true kernel in parenthesis.

A big drawback here is that the computation of sufficiently large dimensional RFs may be very costly, especially when d is large as well. This is a regime where the OPU outperforms CPU and GPU by a large margin (see Figure 4.3) since its computation time is invariant to d and D.

In general, the simulated as well as the physical optical RFs yield similar performances as the RBF Fourier RFs on the provided convolutional data.

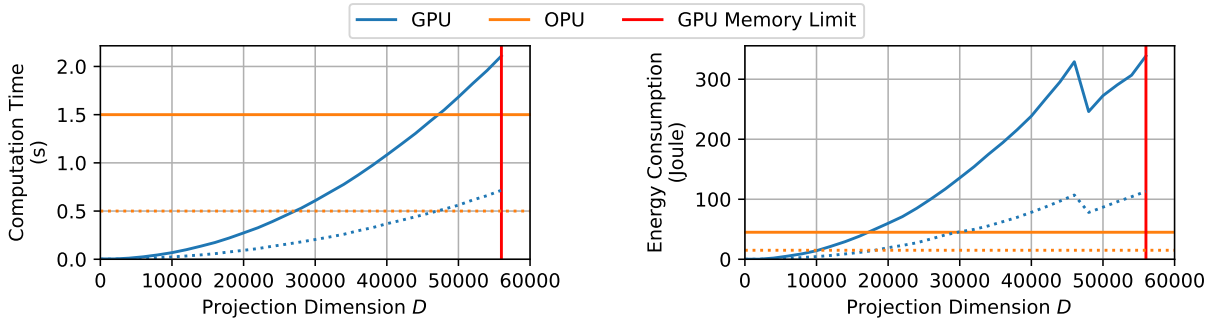


FIGURE 4.3 – Time and energy spent for computing a matrix multiplication $(n, D) \times (D, D)$. The batchsize n is 3000 (solid line) or 1000 (dotted). The curves cross each other at the same D independent from n. We verified more precisely that time and energy are linear with n for both OPU and GPU (experiments were run on a NVIDIA P100).

4.4.3 Projection time and energy consumption

The main advantage of the OPU compared to a traditional CPU/GPU setup is that the OPU takes a constant time for computing RFs of arbitrary dimension D (up to $D = 10^6$ on current hardware) for a single input. Moreover, its power consumption stays below 30 W independently of the workload. Figure 4.3 shows the computation time and the energy consumption over time for GPU and OPU for different projection dimensions D. In both cases, the time and energy spending do not include matrix building and loading. For the GPU, only the calls to the PyTorch function `torch.matmul` are measured and energy consumption is the integration over time of

power values given by the `nvidia-smi` command.
For the OPU, the energy consumption is constant w.r.t. D and equal to 45 Joules (30 W multiplied by 1.5 seconds). The GPU computation time and energy consumption are monotonically increasing except for an irregular energy development between $D = 45\,000$ and $D = 56\,000$. This exact irregularity was observed throughout all simulations we performed and can most likely be attributed to an optimization routine that the GPU carries out internally. The GPU consumes more than 10 times as much energy as the OPU for $D = 58\,000$ (GPU memory limit). The GPU starts to use more energy than the OPU from $D = 18\,000$. The exact crossover points may change in future hardware versions. The relevant point we make here is that the OPU has a better scalability in D with respect to computation time and energy consumption.

Conclusion

The increasing size of available data and the benefit of working in high-dimensional spaces led to an emerging need for dedicated hardware. GPUs have been used with great success to accelerate algebraic computations for kernel methods and deep learning. Yet, they rely on finite memory, consume large amounts of energy and are very expensive.
In contrast, the OPU is a scalable memory-less hardware with reduced power consumption. In this Chapter, we showed that optical RFs are useful in their natural form and can be modified to yield more flexible kernels. In the future, algorithms should be developed to deal with large-scale RFs, and other classes of kernels and applications should be obtained using optical RFs.

4.5 Appendix

4.5.1 Main proof of Theorem 4.3.1

Proof. Thanks to the rotational invariance of the complex Gaussian random vectors $\boldsymbol{u}^{(k)}$, we can fix $\boldsymbol{x} = \|\boldsymbol{x}\|\mathbf{e_1}$ and $\boldsymbol{y} = \|\boldsymbol{y}\|(\cos\theta\mathbf{e_1} + \sin\theta\mathbf{e_2})$. θ represents the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$ and $\cos(\theta) = \frac{\boldsymbol{x}^\top \boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}$. Let $\mathbf{e_i}^\top \boldsymbol{u} = u_i \sim \mathcal{CN}(0,1)$, $i = 1, 2$.
Thus the kernel function becomes :

$$k_2(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2 \int |u_1|^2 |u_1\cos\theta + u_2\sin\theta|^2 \mu(\boldsymbol{u}) d\boldsymbol{u}. \tag{4.9}$$

We then expand the quadratic forms and compute the resulting Gaussian integrals :

$$\begin{aligned}
\frac{1}{\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2} k_2(\boldsymbol{x}, \boldsymbol{y}) &= \int |u_1|^2 |u_1\cos\theta + u_2\sin\theta|^2 \frac{1}{\pi^2} e^{-\frac{\|\boldsymbol{u}\|^2}{2}} d\boldsymbol{u} \\
&= \int |u_1|^2 |u_1\cos\theta + u_2\sin\theta|^2 \frac{1}{\pi^2} e^{-\frac{(|u_1|^2+|u_2|^2)}{2}} d\boldsymbol{u} \\
&= \int \Big(|u_1|^4 \cos^2\theta + |u_1|^2|u_2|^2 \sin^2\theta + 2|u_1|^2 \mathrm{Re}(\overline{u}_1 u_2)\cos\theta\sin\theta \Big) \\
&\quad \frac{1}{\pi^2} e^{-\frac{|u_1|^2+|u_2|^2}{2}} d\boldsymbol{u}.
\end{aligned}$$

The third term in the parenthesis is odd in u_2, so the integral of this term vanishes. Let's remark that if $u \sim \mathcal{CN}(0, \sigma^2)$ then $\mathcal{I}m(u), \mathcal{R}e(u) \sim \mathcal{N}(0, \sigma^{*2} = \frac{1}{2}\sigma^2)$. The two other terms can be computed easily using the moments of the Gaussian distributions (the moment of order 2 of a complex Gaussian random variable is $2\Gamma(2)\sigma^{*2} = 2\sigma^{*2}$, the moment of order 4 is $2^2\Gamma(3)\sigma^{*4} = 8\sigma^{*4}$ where σ^{*2} is the variance of the real and the imaginary part of $u_{1,2}$) .

$$\begin{aligned} k_2(\boldsymbol{x}, \boldsymbol{y}) &= 8\sigma^{*4}\cos^2\theta + 4\sigma^{*4}\sin^2\theta \\ &= 4\sigma^{*4}\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2(2\cos^2\theta + \sin^2\theta) \\ &= 4\sigma^{*4}\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2(1 + \cos^2\theta) \\ &= 4\sigma^{*4}\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2 + 4\sigma^{*4}(\boldsymbol{x}^\top\boldsymbol{y})^2 \\ &= \|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2(1 + \cos^2\theta) \quad \text{if} \quad \sigma^{*2} = \frac{1}{2}. \end{aligned}$$

□

4.5.2 Alternative proof of Theorem 4.3.1

The following is an alternative derivation of Theorem 4.3.1 that breaks the complex random projection into its real and imaginary parts.

Proof. We can rewrite Equation 4.2 as :

$$\boldsymbol{\phi}(\boldsymbol{x}) = \frac{1}{\sqrt{D}}|\mathbf{U}\boldsymbol{x}|^2 = \frac{1}{\sqrt{D}}\Big((\boldsymbol{A}\boldsymbol{x})^2 + (\boldsymbol{B}\boldsymbol{x})^2\Big), \tag{4.10}$$

where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{D\times d}$ are the real and imaginary parts of the complex matrix $\boldsymbol{u} \in \mathbb{C}^{D\times d}$. The elements of $\mathbf{A}$ and $\mathbf{B}$ are i.i.d. draws from a zero-centered Gaussian distribution with variance σ^{*2}.
Now we rewrite the kernel as :

$$\begin{aligned} k_2(\boldsymbol{x}, \boldsymbol{y}) &= \mathbb{E}\Big[|\mathbf{U}\boldsymbol{x}|^2|\mathbf{U}\boldsymbol{y}|^2\Big] = \mathbb{E}\Big[\Big((\boldsymbol{a}^\top\boldsymbol{x})^2 + (\boldsymbol{b}^\top\boldsymbol{x})^2\Big)\Big((\boldsymbol{a}^\top\boldsymbol{y})^2 + (\boldsymbol{b}^\top\boldsymbol{y})^2\Big)\Big] \\ &= \mathbb{E}\Big[(\boldsymbol{a}^\top\boldsymbol{x})^2(\boldsymbol{a}^\top\boldsymbol{y})^2\Big] + \mathbb{E}\Big[(\boldsymbol{b}^\top\boldsymbol{x})^2(\boldsymbol{b}^\top\boldsymbol{y})^2\Big] + \mathbb{E}\Big[(\boldsymbol{a}^\top\boldsymbol{x})^2(\boldsymbol{b}^\top\boldsymbol{y})^2\Big] + \mathbb{E}\Big[(\boldsymbol{b}^\top\boldsymbol{x})^2(\boldsymbol{a}^\top\boldsymbol{y})^2\Big] \\ &= 2\Big(\underbrace{\mathbb{E}\Big[(\boldsymbol{a}^\top\boldsymbol{x})^2(\boldsymbol{a}^\top\boldsymbol{y})^2\Big]}_{(1)} + \underbrace{\mathbb{E}\Big[(\boldsymbol{a}^\top\boldsymbol{x})^2(\boldsymbol{b}^\top\boldsymbol{y})^2\Big]}_{(2)}\Big). \end{aligned} \tag{4.11}$$

Term (1) in Equation 4.11 can be seen as the expectation of the product of two quadratic forms $\mathbb{E}[Q_1(\boldsymbol{a})Q_2(\boldsymbol{a})]$ where $Q_1(\boldsymbol{a}) = \boldsymbol{a}^\top\boldsymbol{x}\boldsymbol{x}^\top\boldsymbol{a}$ and $Q_2(\boldsymbol{a}) = \boldsymbol{a}^\top\boldsymbol{y}\boldsymbol{y}^\top\boldsymbol{a}$. Expectations of products of quadratic forms in normal random variables are well-studied by (Magnus, 1978) [2] and others. Using their result, we can immediately solve Term (1) :

2. The moments of products of quadratic forms in normal variables, Jan R. Magnus, statistica neerlandica, Vol. 32, 1978

$$\underbrace{\mathbb{E}\Big[(\boldsymbol{a}^\top\boldsymbol{x})^2(\boldsymbol{a}^\top\boldsymbol{y})^2\Big]}_{(1)} = \sigma^{*4}\Big(\operatorname{tr}\{\boldsymbol{x}\boldsymbol{x}^\top\}\operatorname{tr}\{\boldsymbol{y}\boldsymbol{y}^\top\} + 2\operatorname{tr}\{\boldsymbol{x}\boldsymbol{x}^\top\boldsymbol{y}\boldsymbol{y}^\top\}\Big) = \sigma^{*4}\Big(\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2 + 2(\boldsymbol{x}^\top\boldsymbol{y})^2\Big). \tag{4.12}$$

Term (2) is easy to solve :

$$\underbrace{\mathbb{E}\Big[(\boldsymbol{a}^\top\boldsymbol{x})^2(\boldsymbol{b}^\top\boldsymbol{y})^2\Big]}_{(2)} = \mathbb{E}\Big[(\boldsymbol{a}^\top\boldsymbol{x})^2\Big]\mathbb{E}\Big[(\boldsymbol{b}^\top\boldsymbol{y})^2\Big] = \Big(\boldsymbol{x}^\top\mathbb{E}\Big[\boldsymbol{a}\boldsymbol{a}^\top\Big]\boldsymbol{x}\Big)\Big(\boldsymbol{y}^\top\mathbb{E}\Big[\boldsymbol{b}\boldsymbol{b}^\top\Big]\boldsymbol{y}\Big) = \sigma^{*4}\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2. \tag{4.13}$$

Inserting both terms into Equation 4.11 yields the desired kernel equation.

□

Although higher degree kernels can be derived in this manner as well, we proceed with the previous method for the following derivations.

4.5.3 Proof of Theorem 4.3.2 : even exponents

If we use the same reasoning as in the proof of Appendix 4.5.1, when the optical random feature is given as in Equation 4.6, i.e. $\boldsymbol{\phi}(\boldsymbol{x}) = \frac{1}{\sqrt{D}}|\mathbf{U}\boldsymbol{x}|^m$, we obtain :

$$\begin{aligned} k_{2s}(\boldsymbol{x},\boldsymbol{y}) &= \int |\boldsymbol{x}^\top\boldsymbol{u}|^m|\boldsymbol{y}^\top\boldsymbol{u}|^m\mu(\boldsymbol{u})d\boldsymbol{u} \\ &= \int \|\boldsymbol{x}\|^m\|\boldsymbol{y}\|^m|u_1|^m|u_1\cos\theta + u_2\sin\theta|^m\mu(\boldsymbol{u})d\boldsymbol{u} \\ &= \|\boldsymbol{x}\|^m\|\boldsymbol{y}\|^m\int |u_1|^m|u_1\cos\theta + u_2\sin\theta|^m\mu(\boldsymbol{u})d\boldsymbol{u}. \end{aligned}$$

Now we focus on the term $A = |u_1\cos\theta + u_2\sin\theta|^{2s}$ (with $m = 2s$) as we focus on even powers :

$$\begin{aligned} A &= (\overline{u}_1\cos\theta + \overline{u}_2\sin\theta)^s(u_1\cos\theta + u_2\sin\theta)^s \\ &= \Big(\sum_{j=0}^{s}\binom{s}{j}(\overline{u}_1\cos\theta)^j(\overline{u}_2\sin\theta)^{s-j}\Big)\Big(\sum_{i=0}^{s}\binom{s}{i}(u_1\cos\theta)^i(u_2\sin\theta)^{s-i}\Big) \\ &= \sum_i\sum_j\binom{s}{j}\binom{s}{i}\overline{u}_1^j u_1^i\overline{u}_2^{s-j}u_2^{s-i}(\cos\theta)^{i+j}(\sin\theta)^{2s-i-j}. \end{aligned}$$

One can notice that $\mathbb{E}[A] = \sum_i\sum_j\binom{s}{j}\binom{s}{i}\mathbb{E}[\overline{u}_1^j u_1^i\overline{u}_2^{s-j}u_2^{s-i}]$ has its cross terms equal to 0 : when $i \neq j$, the expectation inside the sum is equal to zero because u_1 and u_2 are i.i.d (their correlation is then equal to 0) or we can see this by rotational invariance (any complex random variable

with a phase uniform between 0 and 2π has a mean equal to zero). Therefore we obtain :

$$\mathbb{E}[A] = \sum_{i=0}^{s} \binom{s}{i}^2 |u_1|^{2i} |\cos\theta|^{2i} |u_2|^{2(s-i)} |\sin\theta|^{2(s-i)}.$$

Using the computation of $\mathbb{E}[A]$, we can therefore deduce the analytical formula for the kernel :

$$\begin{aligned}
\frac{k_{2s}(\boldsymbol{x},\boldsymbol{y})}{\|\boldsymbol{x}\|^m\|\boldsymbol{y}\|^m} &= \int |u_1|^{2s} \sum_{i=0}^{s} \binom{s}{i}^2 |u_1|^{2i} |\cos\theta|^{2i} |u_2|^{2(s-i)} |\sin\theta|^{2(s-i)} \mu(\boldsymbol{u}) d\boldsymbol{u} \\
&= \sum_{i=0}^{s} \binom{s}{i}^2 \mathbb{E}\Big[|u_1|^{2(s+i)} |\cos\theta|^{2i}\Big]\ \mathbb{E}\Big[|u_2|^{2(s-i)} |\sin\theta|^{2(s-i)}\Big] \\
&= \sum_{i=0}^{s} \binom{s}{i}^2 |\cos\theta|^{2i}\, 2^{s+i} \sigma^{*2(s+i)}\, \Gamma(s+i+1)\, |\sin\theta|^{2(s-i)}\, 2^{s-i} \sigma^{*2(s-i)}\, \Gamma(s-i+1) \\
&= \sum_{i=0}^{s} \binom{s}{i}^2 2^{2s} \sigma^{*4s}\, |\cos\theta|^{2i}\, |\sin\theta|^{2(s-i)} \Gamma(s+i+1)\, \Gamma(s-i+1).
\end{aligned}$$

We can simplify this formula, by noticing that $\sin^2\theta = 1 - \cos^2\theta$. The latter formula becomes :

$$\frac{k_{2s}(\boldsymbol{x},\boldsymbol{y})}{\|\boldsymbol{x}\|^m\|\boldsymbol{y}\|^m} = \sum_{i=0}^{s} \binom{s}{i}^2 |\cos\theta|^{2i}\, (1-\cos^2\theta)^{(s-i)}\, 2^{2s} \sigma^{*4s}\, \Gamma(s+i+1)\, \Gamma(s-i+1).$$

The new term can be expanded using a binomial expansion : $(1-\cos^2\theta)^{(s-i)} = \sum_{t=0}^{s-i} (-\cos^2\theta)^t$, leading to

$$\frac{k_{2s}(\boldsymbol{x},\boldsymbol{y})}{\|\boldsymbol{x}\|^m\|\boldsymbol{y}\|^m} = \sum_{i=0}^{s} \sum_{t=0}^{s-i} \binom{s}{i}^2 (s+i)!(s-i)! \binom{s-i}{t} (-1)^t \cos^{2(i+t)}\theta.$$

Using the change of variable $a = i + t$ and keeping $i = i$, we obtain :

$$\frac{k_{2s}(\boldsymbol{x},\boldsymbol{y})}{\|\boldsymbol{x}\|^m\|\boldsymbol{y}\|^m} = \sum_{a=0}^{s} \Big[\sum_{i=0}^{a} \binom{s}{i}^2 (s+i)!(s-i)! \binom{s-i}{a-i} (-1)^{a-i} \Big] \cos^{2a}\theta.$$

We focus on the term between the brackets that we will call T_a :

$$\begin{aligned}
T_a &= \sum_{i=0}^{a} \frac{s!^2}{(s-i)!^2 i!^2} (s+i)!(s-i)! \frac{(s-i)!}{(a-i)!(s-a)!} (-1)^{a-i} \\
&= \sum_{i=0}^{a} (s!)^2 \frac{(s+i)!}{(i!)^2 (a-i)!(s-a)!} (-1)^{a-i} \\
&= (s!)^2 \binom{s}{a} (-1)^a \sum_{i=0}^{a} \binom{a}{i} \binom{s+i}{i} (-1)^i.
\end{aligned}$$

Using the upper negation ($\binom{a}{b} = (-1)^b \binom{b-a-1}{b}$) so here $\binom{s+i}{i} = \binom{i-i-s-1}{i}(-1)^i = \binom{-s-1}{i}(-1)^i$

leads to

$$T_a = (s!)^2 \binom{s}{a} (-1)^a \sum_{i=0}^{a} \binom{a}{i} \binom{-s-1}{i} \tag{4.14}$$

$$= (s!)^2 \binom{s}{a} (-1)^a \sum_{i=0}^{a} \binom{a}{i} \binom{-s-1}{-s-1-i}. \tag{4.15}$$

Now we can use the Vandermonde identity $\sum_{i=0}^{n} \binom{a}{i}\binom{b}{n-i} = \binom{a+b}{n}$, yielding :

$$T_a = (s!)^2 \binom{s}{a} (-1)^a \binom{a-s-1}{-s-1} = (s!)^2 \binom{s}{a} (-1)^a \binom{a-s-1}{a} \tag{4.16}$$

$$= (s!)^2 \binom{s}{a} \binom{s}{a} = (s!)^2 \binom{s}{a}^2. \tag{4.17}$$

where in the last line we used again the upper negation formula.
This leads to the desired result for $m = 2s$

$$k_{2s}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x}\|^m \|\boldsymbol{y}\|^m \sum_{i=0}^{s} (s!)^2 \binom{s}{i}^2 \cos^{2i}\theta.$$

4.5.4 Convergence properties

For simplicity, we will consider random variables sampled from the normal distribution and not the complex normal one. We will be using the following lemma for proving the convergence in probability of the estimator of the kernel generated using optical random features toward the real kernel :

Lemma 4.5.1. *(Bernstein-type inequality [Vershynin, 2018]) Let $X_1, ..., X_D$ be independent centered sub-exponential random variables, and let $a = (a_1, ..., a_D) \in \mathbb{R}^D$. Then, for every $t \geq 0$, we have :*

$$\mathbb{P}\left\{ \left| \sum_{i=1}^{D} a_i X_i \right| \geq t \right\} \leq 2 \exp\left[- c \min\left(\frac{t^2}{K^2 \|a\|_2^2}, \frac{t}{K \|a\|_\infty} \right) \right] \tag{4.18}$$

with c an absolute constant and $K = \max_i \|X_i\|_{\psi_1}$ ($\|X_i\|_{\psi_1} = \sup_{p \geq 1} p^{-1} (\mathbb{E}|X|^p)^{1/p}$ being the sub-exponential norm of X_i).

Let's start with the case when the exponent is $m = 1$.
We know that $\mathbb{E}[\boldsymbol{\phi}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{y})] = k_1(\boldsymbol{x}, \boldsymbol{y})$ so we can obtain centered random variables by doing :

$$\boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{y}) - k_1(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{y}) - \mathbb{E}[\boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{y})] \tag{4.19}$$

$$= \frac{1}{D} \sum_{i=1}^{D} \left[|\boldsymbol{x}^\top \boldsymbol{u}^{(i)}| \, |\boldsymbol{y}^\top \boldsymbol{u}^{(i)}| - k_1(\boldsymbol{x}, \boldsymbol{y}) \right] \tag{4.20}$$

$$\geq \sum_{i=1}^{D} a_i X_i, \tag{4.21}$$

where we have $a_i = \frac{1}{D}$ and $X_i := |\boldsymbol{x}^\top \mathbf{V}^{(i)} \boldsymbol{y}| - k_1(\boldsymbol{x}, \boldsymbol{y})$ (with $\mathbf{V}^{(i)} = \boldsymbol{u}^{(i)} \boldsymbol{u}^{(i)\top}$).
The matrix $\mathbf{V}^{(i)}$ has elements behaving sub-exponentially as it is the inner product of two sub-Gaussian vectors. Moreover, we have $X_i = |\sum_{j,k} x_j V_{jk}^{(i)} y_k| - k_1(\boldsymbol{x}, \boldsymbol{y})$, so by using lemma 4.5.1, we can deduce that the sum of independent sub-exponential random variables (here the $V_{jk}^{(i)}$) is still sub-exponential. So we can conclude that X_i is a centered sub-exponential random variable.
By applying the Bernstein-type inequality of lemma 4.5.1, we obtain for the feature map of exponent 1 :

$$\mathbb{P}\Big\{\Big|\boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{y}) - k_1(\boldsymbol{x}, \boldsymbol{y})\Big| \geq t\Big\} \leq 2\exp\Big[-c\min\Big(\frac{t^2 D}{K^2}, \frac{tD}{K}\Big)\Big], \tag{4.22}$$

So the convergence of the estimator toward the kernel k_1 is sub-exponential.

Now for the more general case when $m > 1$, we have $\boldsymbol{\phi}(\boldsymbol{x}) = \frac{1}{\sqrt{D}}|\boldsymbol{u}\boldsymbol{x}|^m$ so we can introduce the quantity $W_i(\boldsymbol{x}, \boldsymbol{y}) = |\boldsymbol{x}^\top V^{(i)} \boldsymbol{y}|$ (which has a sub-exponential behavior) such that :

$$\mathbb{P}\Big\{\frac{1}{D}\sum_{i=1}^{D} W_i(\boldsymbol{x}, \boldsymbol{y})^m - k_m(\boldsymbol{x}, \boldsymbol{y}) \geq t\Big\} = \mathbb{P}\Big\{\frac{1}{D}\sum_{i=1}^{D} W_i(\boldsymbol{x}, \boldsymbol{y})^m \geq t + k_m(\boldsymbol{x}, \boldsymbol{y})\Big\} \tag{4.23}$$

$$= \mathbb{P}\Big\{\sum_{i=1}^{D} W_i(\boldsymbol{x}, \boldsymbol{y})^m \geq Dt + Dk_m(\boldsymbol{x}, \boldsymbol{y})\Big\} \tag{4.24}$$

$$\leq \mathbb{P}\Big\{\sum_{i=1}^{D} W_i(\boldsymbol{x}, \boldsymbol{y})^m \geq Dt + k_m(\boldsymbol{x}, \boldsymbol{y})\Big\} \quad (\text{as } D \geq 1) \tag{4.25}$$

$$\leq D\,\mathbb{P}\Big\{W_1(\boldsymbol{x}, \boldsymbol{y}) \geq (Dt + k_m(\boldsymbol{x}, \boldsymbol{y}))^{1/m}\Big\} \quad (\text{Union bound}) \tag{4.26}$$

$$\leq D\,\mathbb{P}\Big\{W_1(\boldsymbol{x}, \boldsymbol{y}) \geq (Dt)^{1/m} + k_m(\boldsymbol{x}, \boldsymbol{y})^{1/m}\Big\} \quad (\text{as } m > 1). \tag{4.27}$$

Now we can use the following Lyapunov inequality. For $0 < s < t$ we have :

$$(\mathbb{E}|X|^s)^{1/s} \leq (\mathbb{E}|X|^t)^{1/t}.$$

However we know that $k_m(\boldsymbol{x}, \boldsymbol{y})^{1/m} = \mathbb{E}[|\boldsymbol{x}^\top \mathbf{V}^{(i)} \boldsymbol{y}|^m]^{1/m}$ and $k_1 = \mathbb{E}[|\boldsymbol{x}^\top \mathbf{V}^{(i)} \boldsymbol{y}|]$, $\forall i \in \{1, \ldots, D\}$, so using Lyapunov inequality we can deduce that for $m > 1$, $k_m(\boldsymbol{x}, \boldsymbol{y})^{1/m} > k_1(\boldsymbol{x}, \boldsymbol{y})$.

Using that result, we can bound Equation 4.27 as follows :

$$\mathbb{P}\Big\{\frac{1}{D}\sum_{i=1}^{D} W_i(\boldsymbol{x},\boldsymbol{y})^m - k_m(\boldsymbol{x},\boldsymbol{y}) \geq t\Big\} \leq D\,\mathbb{P}\Big\{W_1(\boldsymbol{x},\boldsymbol{y}) \geq (Dt)^{1/m} + k_m(\boldsymbol{x},\boldsymbol{y})^{1/m}\Big\} \tag{4.28}$$

$$\leq D\,\mathbb{P}\Big\{W_1(\boldsymbol{x},\boldsymbol{y}) \geq (Dt)^{1/m} + k_1(\mathbf{x},\mathbf{y})\Big\} \tag{4.29}$$

$$\leq D\,\mathbb{P}\Big\{W_1(\boldsymbol{x},\boldsymbol{y}) - k_1(\boldsymbol{x},\boldsymbol{y}) \geq (Dt)^{1/m}\Big\}. \tag{4.30}$$

Using the fact that W_1 is sub-exponential and its expectation is k_1, then $W_1 - k_1(\boldsymbol{x},\boldsymbol{y})$ is a centered random variable. It follows as a conclusion :

$$\mathbb{P}\Big\{\frac{1}{D}\sum_{i=1}^{D} |\boldsymbol{x}^\top \mathbf{V}^{(i)}\boldsymbol{y}|^m - k_m(\boldsymbol{x},\boldsymbol{y}) \geq t\Big\} \leq D\,\exp(-C(Dt)^{1/m}), \tag{4.31}$$

with C being a positive absolute constant. It is not a tight bound but it gives us a behavior in the scaling of the tails which approximately corresponds to $\exp(-(Dt)^{1/m})$.
We can conclude for these convergence rates that the higher the exponent m of the feature map is, the slower is the convergence of the estimator toward the real kernel. It has been noticed experimentally in Figure 4.2 where the convergence of the estimator toward the real kernel for exponent $m = 4$ is slower than the one for $m = 2$.

4.5.5 Extended experimental description

Data encoding. The current version of the OPU only supports binary inputs. There are different ways to obtain a binary encoding of the data. We apply a simple threshold binarizer to each feature j of a datapoint $\boldsymbol{x}_i$:

$$T(x_{ij}) = \begin{cases} 1 & \text{if} \quad x_{ij} > \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{4.32}$$

The optimal threshold θ is determined for every dataset individually such that it maximizes the accuracy on a held-out validation set. Despite the drastic reduction from 32-bit floating point precision to 2 bits, the generalization error drops only by a small amount. The drop is bigger for the convolutional features than for the Fashion-MNIST data.
Hyperparameter search. We carry out an extensive hyperparameter search for every dataset (including each convolutional feature set). The hyperparameters are optimized for every kernel individually using its random feature approximation at $D = 10^4$. A thorough hyperparameter search for every feature dimension as well as for the full kernels would be too expensive. Therefore, the hyperparameters are kept the same for different degrees of kernel approximation.
The optimal set of hyperparameters was found with a grid-search on a held-out validation set. For every kernel, we optimized the feature scale as well as the regularization strength alpha. For the linear and the OPU kernel a bias term that can be appended to the original features was added. For the RBF kernel the third hyperparameter dimension is the gamma/lengthscale

parameter.
The scale and alpha parameters are optimized on a $\log_{10}$-scale that depends on each kernel. The gamma parameter is optimized on a $\log_2$-scale for Fashion-MNIST ; for each convolutional feature set it is found by trying out all values around the maximum non-zero decimal position of the heuristic $\gamma = 1/d$ where d is the original feature dimension. The bias parameter is determined on a log-scale depending on the degree of the kernel.
Figure 4.4 shows an example hyperparameter grid for simulated optical RFs of degree 2 applied to Fashion-MNIST. Brighter colors correspond to higher validation scores.

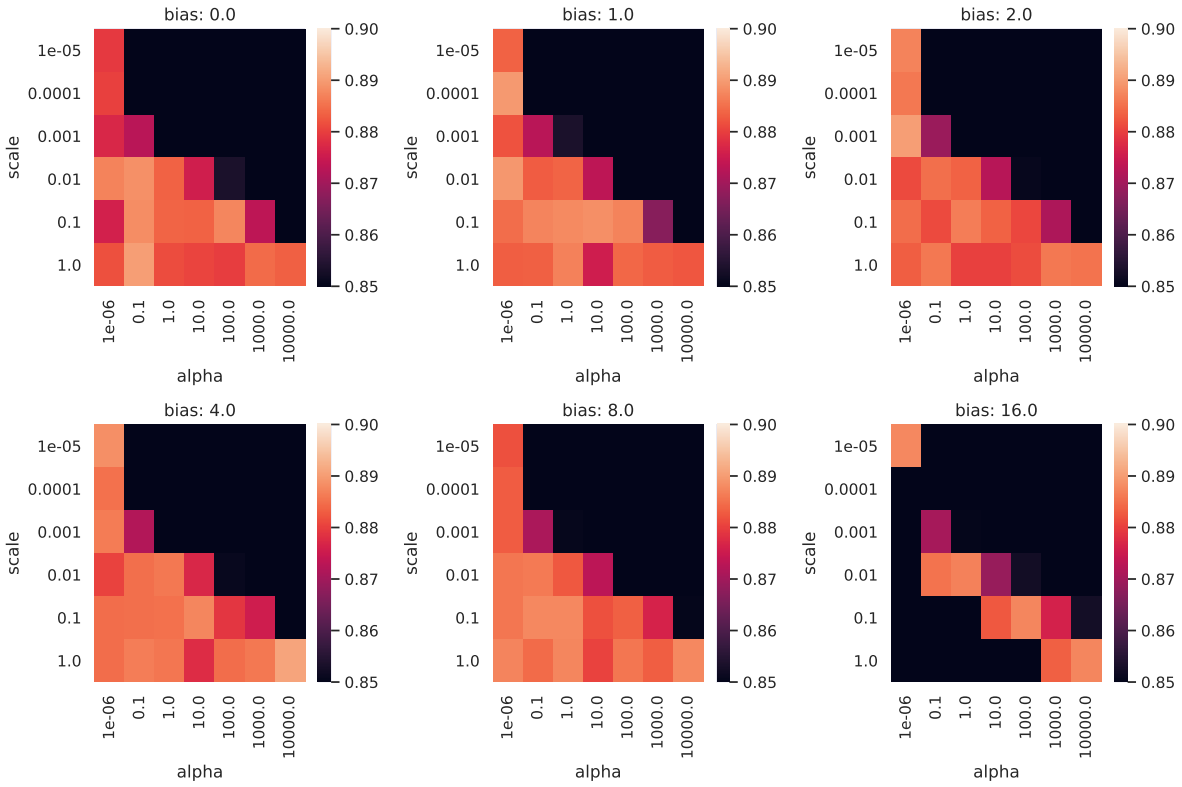


FIGURE 4.4 – Example hyperparameter grid for simulated optical RFs of degree 2 applied to Fashion-MNIST. It is easy to see that the upper triangular part of each grid matrix is a result of over-regularization (α is too large compared to the respective feature scale). Increasing the bias too much also leads to a degrading performance. The optimal hyperparameters are found in the lower-triangular part of one of the hyperparameter grids.

Solvers for linear systems in Ridge Regression. Ridge Regression can be solved either in its primal or dual form. In either case the solution is found by solving a linear system of equations of the form $\mathbf{A}\boldsymbol{x} = \mathbf{b}$. In the primal form we have $\mathbf{A} = (\boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}) + \alpha\mathbf{I})$ and $\mathbf{b} = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{y}$, whereas in the dual form $\mathbf{A} = (\mathbf{K} + \alpha\mathbf{I})$ and $\mathbf{b} = \boldsymbol{y}$. $\boldsymbol{\phi}(\boldsymbol{x})$ is the random projection of the training data. $\boldsymbol{y}$ are the one-hot encoded regression labels ($+1$ is used for the positive and -1 for the negative class). $\mathbf{K}$ is the n-by-n kernel matrix for the training data.
Solving the linear system has computational complexity $\mathcal{O}(D^3)$ or $\mathcal{O}(n^3)$ for the primal and dual form respectively. D is the projection dimension and n the number of datapoints.
Cholesky solvers turned out to work well for primal linear systems up to $D = 10^4$. For higher

dimensions as well as for the dual form (computation of exact kernels) we used the conjugate gradients method.
For the computation of large matrix products as well as conjugate gradients, we developed a memory-efficient method that makes use of multiple GPUs in order to compute partial results. This way stochastic methods were avoided and exact solutions for the linear systems could be obtained.
One issue that arose during the experiments is that solving linear systems for optical RFs using GPUs worsened the conditioning of the matrix $\mathbf{A}$ due to numerical issues. A workaround was to increase α which led to slightly worse test errors.
In practice, we therefore recommend stochastic gradient descent based algorithms that optimize a quadratic regression loss. These allow to work with large-dimensional features while requiring much less memory and giving more numerical stability. Our method is only intended for theoretical comparison and should be used when exact results are needed.

Chapter 5

Adversarial Robustness by Design using OPUs and Direct Feedback Alignment

This chapter is based on [Cappelli, 2021a ; Cappelli, 2021b].

Robustness to adversarial attacks is typically obtained through expensive adversarial training with Projected Gradient Descent or using more expensive and complex training schemes. In this chapter, we introduce a new defense mechanism against adversarial attacks, represented as a simple defense block relying on the use of the Optical Processing Unit, and a training step performed with Direct Feedback Alignment. This defense block is placed just before the classifier of the neural network to defend. In the first part of this chapter, we show that training from scratch a network protected by our defense yields robustness against standard white-box, black-box and transfer attacks. In the second part, we introduce ROPUST, our defense block placed in already robust pre-trained models to further increase their robustness, at no cost in natural accuracy. We test our method on nine different models against four attacks in RobustBench, consistently improving over state-of-the-art performance. In both cases, we perform ablation studies of our defense and demonstrate that phase retrieval attacks are inefficient against it.

5.1 Introduction

Neural networks are sensitive to small, imperceptible to humans, perturbations of their inputs that can cause state-of-the-art classifiers to completely fail [Goodfellow, 2015]. As deep learning models are deployed in real-world applications, guaranteeing their robustness to malicious actors becomes increasingly important : for instance, an adversarial image could evade automated content filtering on social networks [Garcelon, 2020]. Adversarial examples therefore threaten the safety and reliability of machine learning models deployed in the wild but because of the sheer number of attack and defense scenarios, true real-world robustness can however be difficult to evaluate [Bubeck, 2019].

Adversarial attacks can be carried out in different frameworks : in the white-box setting, the attacker has full access to the model, while black-box attacks only rely on queries. It is also possible to craft an attack on a different model and transfer it to the model targeted [Papernot,

2016]. On another hand, standardized benchmarks, such as RobustBench [Croce, 2020a] using AutoAttack [Croce, 2020d], have helped better evaluate progress in the field.
The development of defense-specific attacks is also crucial [Tramèr, 2019]. There is no universal defense, and state-of-the-art techniques often come with a large computational cost, as well as reduced natural accuracy [Tsipras, 2019]. To date, one of the most effective defense and basic techniques remains adversarial training with Projected Gradient Descent (PGD) [Madry, 2018b]. A more advanced adversarial training of a model can be resource-consuming, but fortunately, robust networks pre-trained with PGD are now widely available.
Some of these defenses rely on obfuscated gradients : the model is designed so that the gradients are unsuitable for attacks, for instance by using non-differentiable layers. However, attackers can choose to alter the network structure, using Backward Pass Differentiable Approximation (BPDA) [Athalye, 2018a], replacing obfuscating layers with well-behaved approximations. Furthermore, approaches relying on obfuscation do not generally provide robustness against transfer and black-box attacks.
In this Chapter, we start by expanding the idea of obfuscated gradients to *obfuscated parameters* : we physically implement a fixed random projection followed by a non-linearity using the Optical Processing Unit (OPU), where only the distribution of the random matrix entries is known and not their values. The defense is showed on Figure 5.1. Even though retrieval is possible, the computational cost becomes quickly prohibitive with increasing dimension, and is limited in precision [Gupta, 2019b ; Gupta, 2020b]. To train layers below our non-differentiable defense, we draw inspiration from Direct Feedback Alignment (DFA) [Nøkland, 2016d] and bypass it in the backward pass. We use a random mapping of the global error to train the layer below the OPU, while the layers further downstream perform backpropagation (BP) from this *synthetic gradient* signal. This comes at no natural accuracy cost.
By training our network from scratch, we find our defense is robust by design against white-box attacks : BPDA is ineffective against obfuscated parameters, and attackers are forced to rely on DFA to attack the network. We develop such DFA attacks, and find them much less effective than BP-based attacks on BP-trained networks, confirming results from [Akrout, 2019a].
We also test models incorporating our defense against black-box and transfer attacks, and find that they are more robust than their vanilla counterparts. As parameter obfuscation alone cannot explain robustness in these settings, we perform an ablation study to identify the mechanism responsible for it. For black-box attacks, the combination of binarization and random projection given by the optical co-processor gives a clear contribution to robustness. For transfer attacks, we find the training method itself produces more robust features. Overall, our hybrid training method in conjunction with our optical layer provide a complete defense.
Motivated by the use of pre-trained robust models as a solid foundation for developing simple and widely applicable defenses that further enhance their robustness, we introduce in a second part, **ROPUST**, a drop-in replacement for the classifier of already robust models by replacing it with the defense block studied in the first part.
We evaluate extensively our method against AutoAttack on nine different models in RobustBench, and consistently improve robust accuracies over the state-of-the-art (Section 5.7 and Figure 5.11).

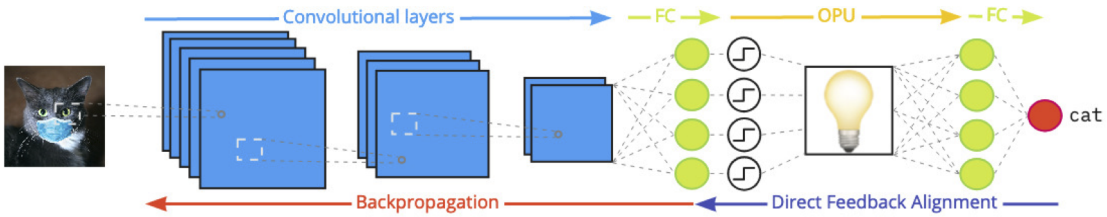


FIGURE 5.1 – A convolutional neural network with an Optical Processing Unit (OPU), our analog defense layer against adversarial attacks. All together, the unknown parameters of the analog operation, the binarization, and the hybrid training method based on Direct Feedback Alignment (DFA) form a defense against white-box, black-box, and transfer attacks.

We perform an ablation study, in Section 5.8, and find that the robustness of our defense against white-box attacks comes from both *parameter obfuscation* and DFA. Surprisingly, we also discover that simply retraining the classifier of a robust model on natural data increases its robustness to square attacks, a phenomenon that warrants further study. Finally, in Section 5.9, we develop a *phase retrieval* attack targeting specifically the parameter obfuscation of our defense, and show that even against state-of-the art phase retrieval techniques, ROPUST achieves fair robustness.

Contributions of the first part. Our study extends gradient obfuscation defenses, introducing a defense based on obfuscation of parameters : this obfuscation is guaranteed by the physical implementation of the defensive layer using the Optical Processing Unit. In the process of evaluating it against black-box and transfer attacks, we find that additional robustness components arise from the binarization and the hybrid training method we use to bypass our non-differentiable defense. In summary :

- We introduce a defense by obfuscated parameters (Section 5.2), implemented by an optical co-processor and develop a hybrid training method inspired by DFA to train models incorporating our defense. This defense comes at no cost in natural accuracy, and no computational cost as all relevant computations are off-loaded to the optical co-processor.
- Our defense is robust by design against FGSM and PGD white-box attacks (Section 5.3.1). To perform white-box attacks against our defense, we introduce hybrid DFA attacks able to bypass our obfuscation, in the spirit of BPDA. However, these are way less effective in creating compelling adversarial examples.
- We find our defense robust against a variety of black-box attacks (NES, bandits and parsimonious). In particular, our defense is significantly more resistant to parsimonious attacks, the strongest of the three evaluated (Section 5.3.2.1). Our defense also provides robustness against transfer attacks (Section 5.3.2.2).
- Since parameters obfuscation cannot explain black-box and transfer robustness, we perform an ablation study to understand the underlying mechanism (Section 5.4). We show that all the elements of our defense are necessary to get robustness to black-box attacks. Surprisingly, for transfer attacks, we find that our training method alone brings robustness.

We perform all experimental benchmarks on CIFAR-10, and CIFAR-100 [Krizhevsky, 2009b]. White-box and transfer of attacks results are obtained with a real optical co-processor as a defense whereas for black-box attacks, a simulated OPU is used for convenience.

Contributions of the second part. We propose to simplify and extend the applicability of the defense previously presented and introduce ROPUST, an universally and easily applicable drop-in replacement for classifiers of *already robust models*. In contrast with the first part, it leverages pre-trained robust models, and achieves state-of-the-art performance. The main contributions are :

- **Simple, universal, and state-of-the-art.** ROPUST can be dropped-in to supplement any robust pre-trained model, replacing its classifier. Fine-tuning the ROPUST classifier is fast and does not require additional changes to the model architecture. This enables any existing architecture and adversarial countermeasure to leverage ROPUST to gain additional robustness, at limited cost. We evaluate on RobustBench, across 9 pre-trained models, against AutoAttack sampling from a pool of 4 attacks. We achieve state-of-the-art performance on the leaderboard, and, in light of our results, we suggest the extension of RobustBench to include obfuscation-based methods.
- **The Square attack mystery.** Performing an ablation study on Square attack [Andriushchenko, 2019], we find that simply retraining from scratch the classifier of a robust model on natural data increases its robustness against it. This phenomenon remains unexplained and occurs even when the original fully connected classification layer is retrained, without using our ROPUST module.
- **Phase retrieval attacks.** Drawing inspiration from the field of phase retrieval, we introduce a new kind of attack against defenses relying on parameter obfuscation, *phase retrieval attacks*. These attacks assume the attacker leverage phase retrieval techniques to retrieve the obfuscated parameters in full, and we show that ROPUST remains robust even against state-of-the-art retrieval methods.

5.2 Preliminaries

5.2.1 Adversarial attacks and defenses

White-box attacks. White-box attacks are adversarial attacks where the attacker is assumed to have full access to the model, including its parameters. In this case, the attacker usually computes a *gradient attack* (e.g. FGSM [Goodfellow, 2015], PGD [Kurakin, 2016 ; Madry, 2018b], or Carlini & Wagner [Carlini, 2017]). These attacks are often fast, effective, and easy to compute. Some defenses obfuscate the gradients to neutralize these attacks : however, it is often possible to build a differentiable approximation (BPDA) to perform gradient-based attacks [Athalye, 2018a], and black-box attacks entirely elude such defenses. In our work, we focus on FGSM and PGD for white-box scenarios (see Figure 5.3).

Black-box attacks. The black-box setting assumes that the attacker has only limited access to the network : for instance, only the label, or the logits for a given input are known. There exist two main categories of black-box attacks. On one hand, gradient estimation attacks [Chen, 2017 ; Ilyas, 2018a ; Ilyas, 2018b] aim to estimate the gradient of the loss with respect to the input to mimic gradient-based attacks. On the other hand, adversarial attacks are transferable [Papernot, 2016] : an attack effective on a given network is likely to also fool another network. More recently, black-box methods based on optimisation tools derived from genetic algorithms [Meunier, 2019 ; Andriushchenko, 2019] and combinatorial optimization [Moon, 2019b] have been introduced. We evaluate our defense against NES and bandits, two gradient-estimation attacks, and parsimonious black-box methods, as well as transfer of attacks.

Defenses Historically, the first defense proposed against attacks was adversarial training [Goodfellow, 2015 ; Madry, 2018b] (i.e. training the neural network through a min-max optimization framework) thus including adversarial robustness as an explicit training objective. Despite its simplicity and lack of theoretical guarantees, adversarial training is still one of the most effective defense against adversarial examples. Theoretically proven defenses also exist, such as randomized smoothing [Lecuyer, 2018 ; Cohen, 2019 ; Pinot, 2019 ; Alexandre Araujo, 2020] or convex relaxation [Wong, 2018a ; Wong, 2018b]. In the literature, numerous defenses do not evaluate their models on attacks adapted to the defense [Tramer, 2020] : especially in the case of gradient obfuscation [Athalye, 2018a], which can result in a false sense of security. In contrast, we evaluate our defense in a wide range of scenarios, and introduce new DFA-based white-box attacks. We also drive a thorough and rigorous ablation study, to better understand the mechanisms underlying the robustness of our defense.
Finally, defense techniques often demand extra computations and reduce natural accuracy. Instead, our defense computations are offloaded to the optical co-processor and the decrease in natural accuracy is minimal, in contrast with adversarial training. More details about adversarial attacks and defenses are given in Section 2.5.1.

5.2.2 The defense

Optical Processing Units. Optical Processing Units (OPU) are photonic co-processors dedicated to efficient large-scale random projections. Assuming an input vector $\boldsymbol{x}$, the OPU computes the following operation using light scattering through a diffusive medium :

$$\boldsymbol{y} = |\mathbf{U}\boldsymbol{x}|^2, \tag{5.1}$$

with $\mathbf{U}$ a *fixed* complex Gaussian random matrix of size up to $10^6 \times 10^6$, which entries are not readily known. In the following, we sometimes refer to $\mathbf{U}$ as the *transmission matrix* (TM). The input $\boldsymbol{x}$ is binary (1 bit – 0/1) and the output $\boldsymbol{y}$ is quantized in 8-bit. While it is possible to simulate an OPU and implement our defense on GPU, this comes with two significant drawbacks : (1) part of our defense relies on $\mathbf{U}$ being obfuscated to the attacker, which is not possible to guarantee on a GPU ; (2) at large sizes, storing $\mathbf{U}$ in GPU memory is costly [Ohana, 2020]. More

details about the OPU are given in Section 2.4.

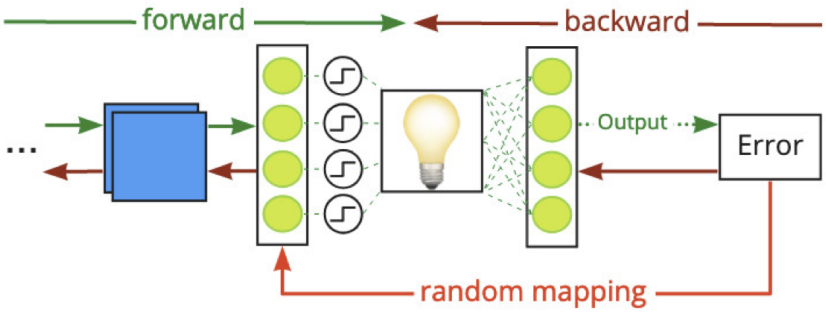


FIGURE 5.2 – The non-differentiable defense layer is bypassed in the backward path by using a random mapping of the error as a synthetic gradient. This signal is then used in the backpropagation update of the previous layers.

Retrieval of the matrix is possible [Gupta, 2019b ; Gupta, 2020b] with direct access to the OPU, but it is computationally expensive for large enough dimensions and relative errors can be as high as 30%. The fastest known method [Gupta, 2020b] for the retrieval of a matrix $\mathbf{U} \in \mathbb{C}^{M \times N}$ relies on the multilateration of anchor signals and has $O(MN \log N)$ time complexity (we develop an attack scenario based on this method in Section 5.9). For $N \sim 10^4$ and $M \sim 10^5$, retrieval with relative error of 32.0% takes 72 minutes. If we wanted to recover the same matrix with a relative error of 8.0%, we would need 19 hours, as decreasing the relative error has a quadratic cost – see Section 5.4.3 in the supplementary for details. The optical system can scale up to $N, M \sim 10^6$: at these dimensions the random matrix alone takes about 8 TB to store, making memory the main constraint in the retrieval. Finally, it is also easily possible to change the entries of the matrix of the optical system to another draw from the same probability distribution. To adapt to this new random matrix, only the classifier has to be fine-tuned : this enables a defense strategy where the random matrix is regularly resampled, preventing malicious actors from having enough time to recover the it.
Accordingly, our defense effectively achieves *parameter obfuscation*, preventing attackers from building a differentiable approximation that can be used to reliably generate adversarial examples. In our work we use an actual optical co-processor for white-box and transfer attacks, and a simulated one for black-box. Note that while we simulate input binarization, we don't simulate output quantization to 8 bits, as we find this quantization to be of little influence.

Direct Feedback Alignment. Because the fixed random parameters implemented by the OPU are unknown, it is impossible to backpropagate through it. We bypass this limitation by training layers upstream of the OPU using Direct Feedback Alignment (DFA) [Nøkland, 2016b]. DFA is an alternative to backpropagation, capable of scaling to modern deep learning tasks and architectures [Launay, 2020a], relying on a random projection of the error as the teaching signal. In a fully connected network, at layer ℓ out of L, neglecting biases, with $\mathbf{W}_\ell$ its weight matrix, f_ℓ its non-linearity, and $\mathbf{h}_\ell$ its activations, the forward pass can be written as $\mathbf{a}_\ell = \mathbf{W}_\ell \mathbf{h}_{\ell-1}, \mathbf{h}_\ell = f_\ell(\mathbf{a}_\ell)$. $\mathbf{h}_0 = \boldsymbol{x}$ is the input data, and $\mathbf{h}_L = f(\mathbf{a}_L) = \hat{\boldsymbol{y}}$ are the predictions. A task-specific cost function $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ is computed to quantify the quality of the predictions with respect to the targets $\boldsymbol{y}$. The

weight updates are obtained through the chain-rule of derivatives :

$$\delta \mathbf{W}_\ell = -\frac{\partial \mathcal{L}}{\partial \mathbf{W}_\ell} = -[(\mathbf{W}_{\ell+1}^T \delta \mathbf{a}_{\ell+1}) \odot f'_\ell(\mathbf{a}_\ell)]\mathbf{h}_{\ell-1}^T, \delta \mathbf{a}_\ell = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_\ell}, \tag{5.2}$$

where $\odot$ is the Hadamard product. With DFA, the gradient signal $\mathbf{W}_{\ell+1}^T \delta \mathbf{a}_{\ell+1}$ of the $(\ell+1)$-th layer is replaced with a random projection of the gradient of the loss at the top layer $\delta \mathbf{a}_y$ –which is the error $\mathbf{e} = \hat{\boldsymbol{y}} - \boldsymbol{y}$ for commonly used losses, such as cross-entropy or mean squared error :

$$\delta \mathbf{W}_\ell = -[(\mathbf{B}_\ell \delta \mathbf{a}_y) \odot f'_\ell(\mathbf{a}_\ell)]\mathbf{h}_{\ell-1}^T, \delta \mathbf{a}_y = \frac{\partial \mathcal{L}}{\partial \mathbf{a}_y}. \tag{5.3}$$

Learning with DFA is enabled by an alignment process, wherein the forward weights learn a configuration enabling DFA to approximate BP updates [Refinetti, 2021]. More details about DFA are given in Section 2.3.

5.3 Experimental results on networks trained from scratch

We place our defense after the convolutional layers of a VGG-16 architecture [Simonyan, 2014b] that we will train from scratch. We call this network *VGG-OPU*. The specific hyperparameters of our model are precised in Section 5.4.3 . The training is performed with the hybrid BP-DFA algorithm discussed in the Section 5.2.2, and shown in Figure 5.1. The code of DFA is taken from [Launay, 2020a]. We consider the CIFAR-10 and CIFAR-100 datasets.
We attack our models using only images that were correctly classified, with the exception of white-box attacks, where we use the full datasets. All the attacks are *untargeted* : we aim to change the classification without any specific target label. Finally, the loss used for computing the attacks is the cross-entropy between the output of the classifier for a given input and its label : $l(\boldsymbol{x}, y) = -\log p_\theta(y|\boldsymbol{x})$. As we consider untargeted attacks, we aim to maximize it.

5.3.1 White-box attacks

We first consider white-box attacks : the attacker has full knowledge of the model and its parameters, and can craft adversarial examples by gradient-based methods. We show that our parameter obfuscation approach makes these attacks significantly less effective, forcing them to rely on imprecise gradient approximations based on DFA or BPDA.
FGSM attacks [Goodfellow, 2015]. We perform the simplest attacks, namely Fast Gradient Sign Method which consists in building the perturbation with the sign of the gradient of the loss.
PGD attacks [Madry, 2018a]. We perform PGD attacks, where the adversarial example $\boldsymbol{x}$ is iteratively optimized with :

$$\boldsymbol{x}^{t+1} = \Pi_{B_\infty(\boldsymbol{x},\epsilon)} \left[\boldsymbol{x}^t + \alpha \text{sign} \left(\nabla_{\boldsymbol{x}} l(\boldsymbol{x}^t, y) \right) \right],$$

where $\Pi_{B_\infty(\boldsymbol{x},\epsilon)}$ is the orthogonal projection on $B_\infty(\boldsymbol{x}, \epsilon) := \{\boldsymbol{x}' : \|\boldsymbol{x}' - \boldsymbol{x}\|_\infty \leq \epsilon\}$ and $\boldsymbol{x}^0 = \boldsymbol{x}$. The quantity ϵ is the strength of the perturbation. As FGSM is weaker than PGD, we will focus

on PGD in this section, and even though FGSM results are presented on Figure 5.3.
To attack networks despite our non-differentiable defense, we either use our hybrid training method to generate attack gradients by skipping our defense, or consider building a BPDA of our defense : the sign function is approximated by a tanh, and the unknown random weights are approximated by another set of random weights.

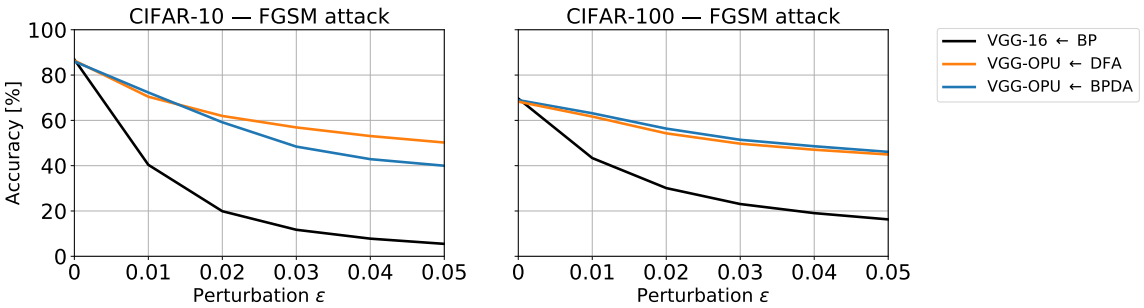


FIGURE 5.3 – Notation : <model> ← <attack gradients>. For example VGG ← BP means that a VGG-16 is attacked with gradients computed with backpropagation. The VGG-OPU model is again more robust than the VGG-16 baseline. BPDA fails again to produce better attacks than our hybrid training method.

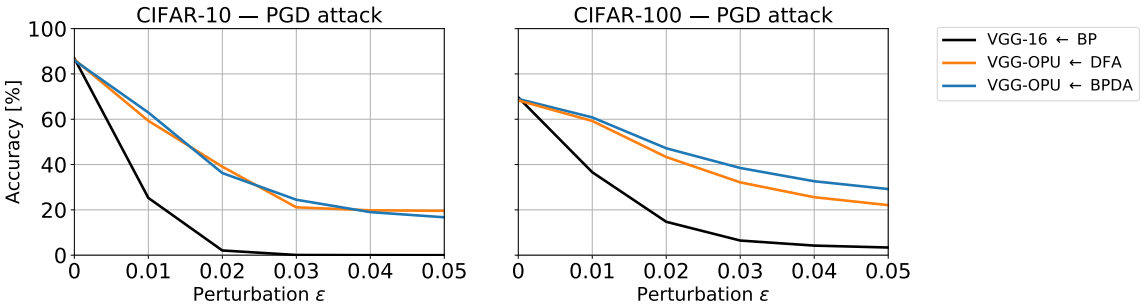


FIGURE 5.4 – Notation : <model> ← <attack gradients>. For example VGG ← BP means that a VGG-16 is attacked with gradients computed with backpropagation. The VGG-OPU model is systematically more robust than the VGG-16 baseline. The failure of BPDA to produce better attacks than our hybrid training method (here abbreviated DFA) confirms that parameters obfuscation enables robustness to white-box attacks by design.

Results Results are shown for FGSM on Figure 5.3 and for PGD on Figure 5.4. We find the VGG-OPU model incorporating our defense performs better than the VGG-16 baseline for any value of ϵ, with gains in accuracy under attack ranging from 20% to 40%. For the largest values of ϵ considered, while the accuracy of the baseline goes to zero, the VGG-OPU model is still performing better than a random guess. These results also show that our obfuscated parameters approach cannot be fooled by a simple BPDA like obfuscated gradients can be : BPDA is here ineffective at finding better attacks than simply bypassing our defense with DFA. Finally, we note the increased robustness does not come at a natural accuracy cost ($\epsilon = 0$).

5.3.2 Black-box attacks

If the obfuscated parameters approach provides robustness by design against white-box attacks, black-box approaches should be not affected, since they do not require knowledge of the weights. However, we find our defense still brings robustness against such attacks.

5.3.2.1 NES, bandits, and parsimonious attacks

Background To further test the robustness of our defense, we perform strong black-box attacks : NES [Ilyas, 2018b], bandits [Ilyas, 2018b], and parsimonious attacks [Moon, 2019a]. NES and bandits attacks consist in efficiently estimating the gradient of the loss with regards to the input $\boldsymbol{x}$:

$$\begin{aligned}\nabla_{\boldsymbol{x}} l(\boldsymbol{x}, y) &\simeq \nabla_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{n}\sim\mathcal{N}(\boldsymbol{x},\sigma^2\mathbf{I})}\left[l(\boldsymbol{n}, y)\right] \\ &\simeq \frac{1}{\sigma N}\sum_{i=1}^{N} \boldsymbol{\delta}_i l(\boldsymbol{x} + \sigma\boldsymbol{\delta}_i, y),\end{aligned}$$

where y is the true label, $\boldsymbol{\delta}_i$ are standard Gaussian random variables and N the sample size of the Monte-Carlo estimation. Bandits attacks are an extension of NES attacks, taking advantage of gradients correlations for close pixels and between gradient steps. To ensure we are not robust only against gradient estimation-based black-box attacks, we also consider parsimonious attacks. This attack is different in nature from the previous two, as it is inspired by combinatorial optimization. It is worth mentioning that this attack is not very sensible to hyperparameters choice [Moon, 2019a]. Parsimonious attacks are also significantly stronger, outperforming NES and bandits approaches.

We measure the Cumulative Success Rates (CSR) in terms of elapsed queries budget. We fix a maximum budget of 15000 queries to the classifier for black-box attacks. This budget is sufficient to reach a plateau in the success rate of the attacks, and to achieve 100% success rate with parsimonious attacks on an undefended baseline. We select attack hyperparameters obtaining

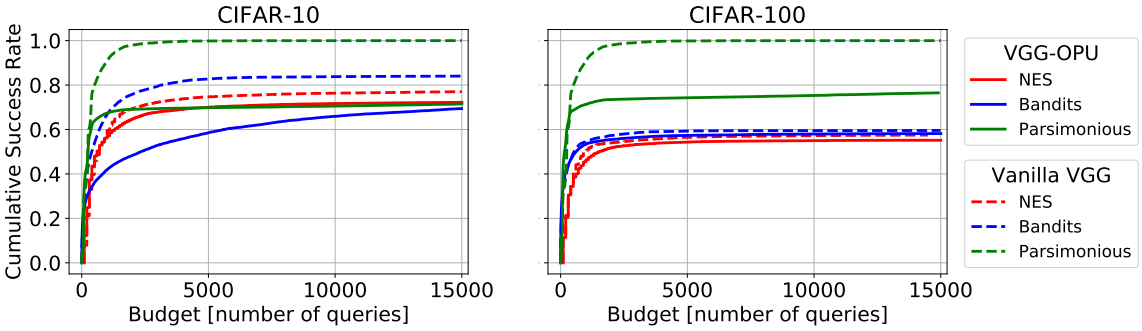


FIGURE 5.5 – Cumulative success rate with respect to the number of queries for different black-box attacks on the CIFAR-10 and CIFAR-100 datasets, on a VGG-16 **(dashed lines)** and a VGG-OPU with a simulated OPU **(plain lines)**. The architectures with our defense perform on par or better than the baseline. The parsimonious attack has the highest success rate, that with our defense drops by 0.3 on CIFAR-10 and 0.2 on CIFAR-100.

the best success rates, as described in Section 5.4.3.

Results Results are shown in Figure 5.5. For the gradient-estimation attacks (NES and bandits), our defense improves robustness with respect to the baseline by decreasing the CSR respectively by 5% and 10 % for the largest budget on CIFAR-10. On CIFAR-100, the improvement in robustness is minimal, of the order of 1% for both attacks. However, our defense shows significant improvement against parsimonious attacks (that largely outperform both NES and bandits), reducing the CSR by 30% on CIFAR-10 and by 24% CIFAR-100. Overall, the best attack on Vanilla VGG reaches 100% CSR on both datasets whereas the best attack on a VGG with our defense reaches only 70% on CIFAR-10 and 76% on CIFAR-100.

5.3.2.2 Transfer attacks

Background Finally, we test the robustness of our defense against transfer attacks. In this scenario, attacks are crafted on a separate source network built by the attacker, similar to the target network. This is a form of black-box attack that does not require queries to the target network.
To evaluate transfer attacks, we first create a test set of well classified samples common to both the source and target network. We then perform attacks on a VGG-16 model on this dataset using FGSM and PGD attacks, building a collection of adversarial examples to transfer to other networks – FGSM results are presented in Figure 5.6.

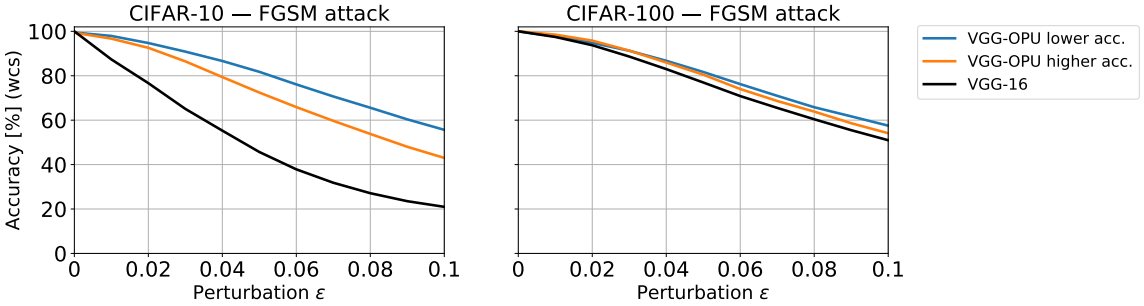


FIGURE 5.6 – Accuracy on a common well-classified set under transfer attacks of increasing strength. Even for FGSM attacks both VGG-OPU networks are more robust than the baseline. Once again we observe a trade-off between robustness and natural accuracy with our defense. Higher/lower accuracies are $\sim$ 85%/80% for CIFAR-10 and $\sim$ 72%/68% for CIFAR-100. The baseline VGG-16 was trained to reach the higher accuracy.

The target networks are a vanilla VGG-16 trained with backpropagation, and a VGG-OPU optimized as follows : we first train a vanilla VGG-16 with our hybrid training method and then place our OPU defense after the trained convolutional stack, finetuning only the classifier layer. We also study two different VGG-OPU models, with identical architectures, but different overall natural accuracy to evaluate if there exists a robustness-accuracy trade-off.

Results The results of this study are shown in Figure 5.6 and 5.7. The VGG-OPU network performs better against transfer attacks for all perturbation strength values. On CIFAR-10,

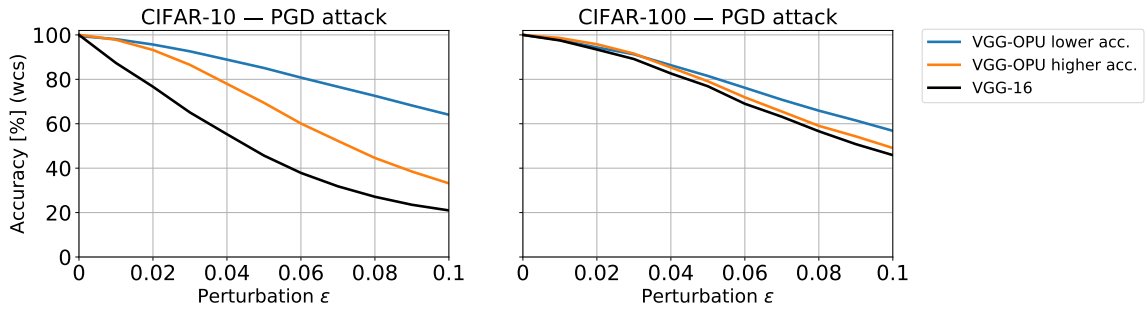


FIGURE 5.7 – Accuracy on a common well-classified set under transfer attacks of increasing strength. Both VGG-OPU networks are more robust than the baseline, and we observe a trade-off between robustness and natural accuracy with our defense. Higher/lower accuracies are $\sim 85\%/80\%$ for CIFAR10 and $\sim 72\%/68\%$ for CIFAR100. The baseline VGG-16 was trained to reach the higher accuracy.

we find our defense to provide significant robustness to transfer, between +15% and +45% of robustness on the well classified evaluation dataset. This gain is smaller on CIFAR-100, between +3% and +10% robustness, where the transfer attack is overall less effective. In either case, we find that there exists an accuracy-robustness trade-off : at the cost of 5% of natural accuracy, robustness can be increased. This makes the defense customizable, allowing to trade some accuracy for robustness.

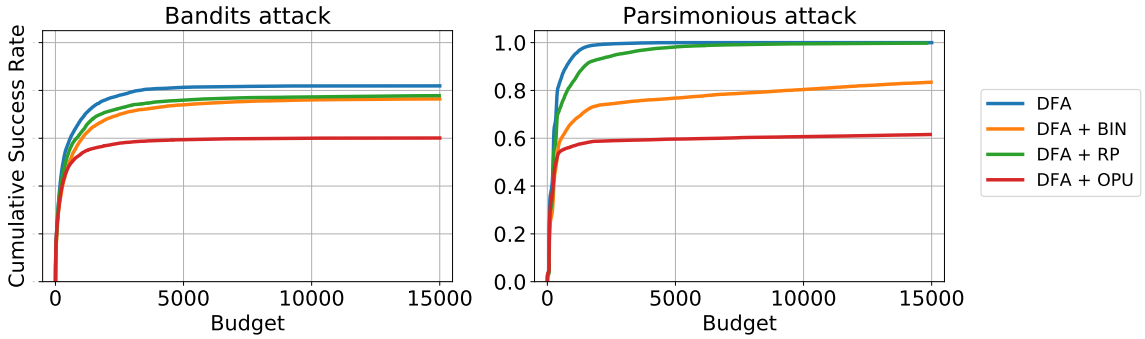
5.4 Beyond obfuscated parameters : understanding black-box and transfer robustness

While obfuscated parameters provide robustness by design against white-box attacks, their use alone cannot explain robustness in black-box and transfer scenarios. We perform ablation studies in these two scenarios to understand the increased robustness. We dissect our defense into three different parts : the hybrid training method involving Direct Feedback Alignment **(DFA)**, the binarization before the optical transform **(BIN)**, and the non-linear random projection of Equation 5.1 performed by the OPU **(RP).** We denote the Optical Processing Unit **(OPU)** as the combination of **BIN + RP.** All the models considered are trained to reach similar test accuracy on CIFAR-10 using the hybrid BP-DFA training, to avoid any robustness-accuracy trade-off effect.

5.4.1 Black-box attacks

We first consider an ablation study on a black-box gradient-estimation bandits attacks, in Figure 5.8.

We notice that DFA alone does not provide robustness against gradient-estimation black-box attacks, but adding either a binary layer (BIN) or a random projection (RP) improves robustness by around 5%. The most striking contribution to the robustness comes from combining the binary layer and the random projection, i.e. the OPU layer, which drops the CSR by 20%, at no



ht

FIGURE 5.8 – Ablation study on the bandits and the parsimonious attack. The robustness of our defense comes from the binarization and random projection steps. In particular, it's the combination of these two mechanisms that leads to a significant increase in robustness.

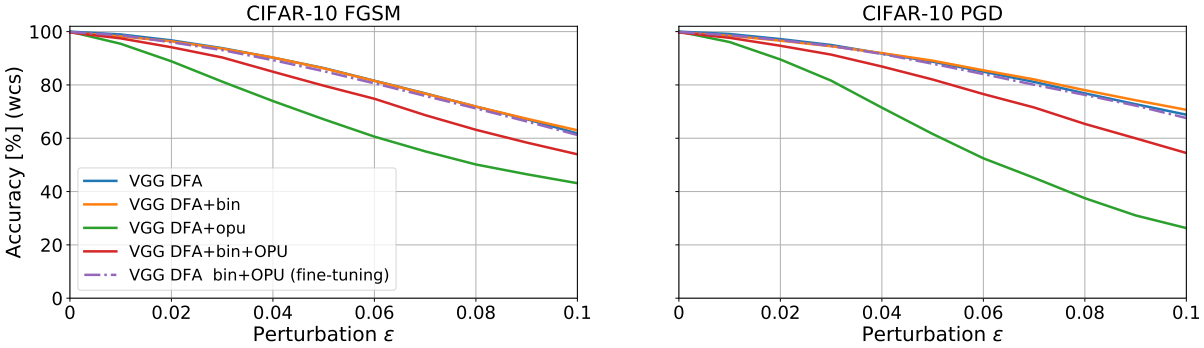
cost in natural accuracy. The failure of DFA to provide robustness against gradient-estimation black-box attack is not surprising : as a training method, it approximates backpropagation with some added noise to the gradients – this is of little effect to the end model in terms of robustness to gradient-estimation attacks. Conversely, the use of a binarization layer and of random projection change the optimization landscape of the model. In particular, the binary layer makes this landscape harder to navigate, even for a black-box optimization method.
We then consider an ablation study on parsimonious attacks, as they are based on combinatorial optimization instead of gradient-estimation. The results are shown in Figure 5.8. We find that the random projection almost doesn't contribute to the robustness. However, binarization has a much stronger effect, decreasing the asymptotic CSR by 20%. Used in conjunction, the random projection and the binarization decrease the CSR by 40%. Overall, this confirms that the combination of all the elements of our defense is necessary to obtain black-box robustness.

5.4.2 Transfer attacks

To complete our ablation study, we seek to understand the source of the robustness against transfer attacks. Results are plotted in Figure 5.9. Surprisingly, at variance with our results on classic black-box attacks, the hybrid training method alone is responsible for the robustness against transfer attacks. In fact, the model with hybrid training and OPU is less robust than the model with hybrid training alone. Nevertheless, we find that the robust features learned by hybrid training are transferable : when fine-tuning a classifier including our optical defense on said features, we find we conserve their robustness. We hypothesize that hybrid training with DFA builds features that find a different optimum in the loss landscape than features derived from BP. Accordingly, attacks on features learned by BP transfer poorly to these DFA features.

5.4.3 Experimental details

Description of our VGG models. The baseline architecture we use is a Vanilla VGG-16. The convolutional stack and the classifier are based on `https://github.com/chengyangfu/pytorch-vgg-cifar10` (configuration D with batchnorm). For CIFAR-100 we simply modify the



ht

FIGURE 5.9 – Ablation study of robustness to transfer attacks. All of the architectures are trained to reach similar accuracy with the BP-DFA algorithm. DFA, a VGG-16 trained with BP-DFA is robust on its own : this proves that the hybrid training algorithm is providing robustness against transfer-attacks. Adding a binary layer is not detrimental to the robustness while adding a random projection decreases it. Adding a binary layer before the random projections achieves a compromise. Finally, we show how it is possible to fine-tune a classifier with an OPU on top of the robust features learned by DFA with no loss in robustness and natural accuracy.

last linear layer to adapt to the higher number of classes.

The Vanilla VGG-16 has three fully connected (fc) layers after the convolutional stack of sizes : fc1 : $512 - 512$, fc2 : $512 - 512$, fc3 : $512 - \text{classes}$ (classes $= 10$ for CIFAR-10 and 100 for CIFAR-100). The VGG-OPU shares the same convolutional stack of the vanilla VGG-16 ; it has however a simulated or real OPU (binary layer followed by a nonlinear random projection) instead of fc2. We have used OPU input size 512 and output size 8000 for white-box attacks, input size 1024 and output size 8000 for transfer attacks and the ablations study, finally input size 2000 and output size 10000 for black-box attacks. Further details on the hyperparameters used to train the models (e.g. learning rate, weight decay) can be found in the code.

Adversarial attacks hyperparameters *White-box attacks* were performed on images normalized in $[-1, 1]$. To create adversarial attack with PGD we used 50 iterations with $\alpha = 0.01$ step size. The PGD and FGSM attacks for transfer experiments have been created using the same parameters.

Black-box attacks in the following were performed on images normalized in $[0, 1]$.

NES attacks were performed using the cross-entropy loss on the L_∞ ball of radius $8/256$. The maximum number of queries is 15000, the number of samples to estimate the gradients is 50 and the size of a batch 1024. We vary the standard deviation σ at values $[0.05, 0.1, 0.5, 1]$. We keep the attack with the σ yielding the best Cumulative Success Rate for each model on each dataset.

Bandits attacks were also performed using the cross-entropy loss on the L_∞ ball of radius $8/256$. The maximum number of queries is 15000 and the number of gradient iterations is kept to 1. We vary the standard deviation σ at values $[0.1, 0.5, 1]$. The online learning rate is set at 0.1, the exploration at 0.1 and the prior size at 16.

Parsimonious attacks were performed with $\epsilon = 8/256$, the number of iterations in local search is 1, the initial block size is 4, the size of a batch is 64 and no hierarchical evaluation was performed.

Retrieval of obfuscated parameters. In the following we provide practical information about the retrieval of the OPU matrix.

Setting With direct access to the host system and the optical co-processor, it is possible to use phase retrieval techniques to retrieve the transmission matrix of the co-processor and hence the obfuscated parameters.

Setup The timings mentioned in this Chapter are obtained on a server with a dual CPU setup, with Intel(R) Xeon(R) Gold 6130 CPU @ 2.10 GHz. Implementing the same algorithm on GPU has only a marginal effect on performance.

Limitations The complexity scales linearly with the output size, $O(n \log n)$ with the input size n, and quadratically with the relative error, that is reducing the relative error by half costs a $4\times$ increase in execution time. However, the main limit of the retrieval algorithm is in terms of memory use and co-processor abilities : to recover a transmission matrix with 10^3 rows/columns, we need to project and process a matrix with 10^5 rows/columns (the calibration signal). First, this procedure rapidly requires large amounts of RAM. Moreover, given the limit of input and output size of the optical system at 10^6, the impossibility to increase the calibration signals further limits the achievable precision of the retrieval.
Finally, it is possible to change the transmission matrix of the optical system, simply by changing some characteristics of the input light (e.g. the wavelength).

5.5 ROPUST : improving robustness of already robust models

5.5.1 Related work

Attacks. Adversarial attacks have been framed in a variety of settings : white-box, where the attacker is assumed to have unlimited access to the model, including its parameters (e.g. FGSM [Goodfellow, 2015], PGD [Madry, 2018b ; Kurakin, 2016], Carlini & Wagner [Carlini, 2017]) ; black-box, assuming only limited access to the network for the attacker, such as the label or logits for a given input, with methods attempting to estimate the gradients [Chen, 2017 ; Ilyas, 2018a ; Ilyas, 2018b], or more recently derived from genetic algorithms [Andriushchenko, 2019 ; Meunier, 2019] and combinatorial optimization [Moon, 2019a] ; transfer attacks, where an attack is crafted on a similar model that is accessible to the attacker, and then applied to the target network [Papernot, 2016]. Automated schemes, such as AutoAttack [Croce, 2020d], have been proposed to autonomously select which attack to perform against a given network, and to automatically tune its hyperparameters.

Defenses. Adversarial training adds adversarial robustness as an explicit training objective [Goodfellow, 2015 ; Madry, 2018b], by incorporating adversarial examples during the training. This has been, and still is, one of the most effective defense against attacks. Repository of

pre-trained robust models have been compiled, such as the RobustBench Model Zoo [1]. Conversely, theoretically grounded defenses have been proposed [Lecuyer, 2018 ; Cohen, 2019 ; Alexandre Araujo, 2020 ; Pinot, 2019 ; Wong, 2018b ; Wong, 2018a], but these fail to match the clean accuracy of state-of-the-art networks, making robustness a trade-off with performance. Many empirical defenses have been criticized for providing a false sense of security [Athalye, 2018b ; Tramèr, 2019], by not evaluating on attacks adapted to the defense.

Obfuscation. Gradient obfuscation, through the use of a non-differentiable activation function, has been proposed as a way to protect against white-box attacks [Papernot, 2017]. However, gradient obfuscation can be easily bypassed by Backward Pass Differentiable Approximation (BPDA) [Athalye, 2018b], where the defense is replaced by an approximated and differentiable version. *Parameter obfuscation* has been proposed with dedicated photonic co-processor in the first part, enforced by the physical properties of said co-processor. However, by itself, this kind of defense falls short of adversarial training.

Fine-tuning and analog computing. Previous work introduced *adversarial fine-tuning* [Jeddi, 2020] : fine-tuning a non-robust model with an adversarial objective. In this work instead we fine-tune a robust model without adversarial training. Additionally, it was shown that robustness improves transfer performance [Salman, 2020] and that robustness transfers across datasets [Shafahi, 2020]. The advantage of non-ideal analog computations in terms of robustness has been investigated in the context of NVM crossbars [Roy, 2020], while we here focus on a photonic technology, readily available to perform computations at scale.

5.6 Methods

5.6.1 Automated adversarial attacks

We evaluate our model against the four attacks implemented in RobustBench : APGD-CE and APGD-T [Croce, 2020d], Square attack [Andriushchenko, 2019], and Fast Adaptive Boundary (FAB) attack [Croce, 2020b]. APGD-CE is a standard PGD where the step size is tuned using the loss trend information, squeezing the best performance out of a limited iterations budget. APGD-T, on top of the step size schedule, substitutes the cross-entropy loss with the Difference of Logits Ratio (DLR) loss, reducing the risk of vanishing gradients. Square attack is based on a random search. Random updates δ are sampled from an attack-norm dependent distribution at each iteration : if they improve the objective function they are kept, otherwise they are discarded. FAB attack aims at finding adversarial samples with minimal distortion with respect to the attack point. With respect to PGD, it does not need to be restarted and it achieves fast good quality results. In RobustBench, using AutoAttack, given a batch of samples, these are first attacked with APGD-CE. Then, the samples that were successfully attacked are discarded, and the remaining ones are attacked with APGD-T. This procedure continues with Square and FAB attack.

1. Accessible at : https://github.com/RobustBench/robustbench.

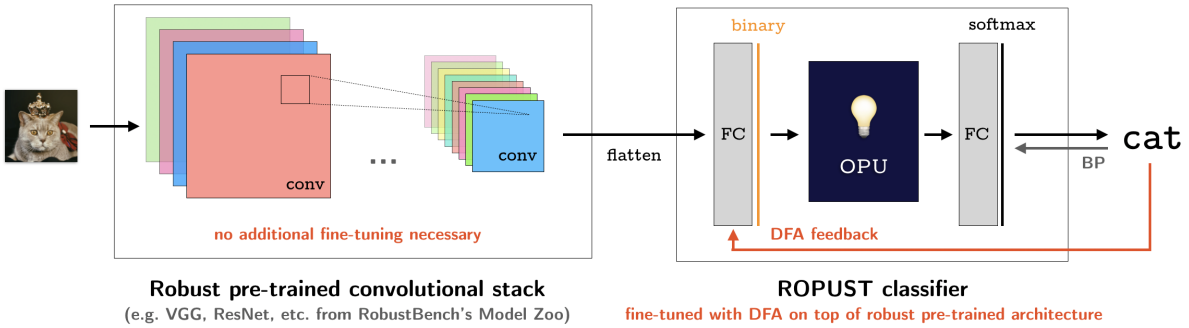


FIGURE 5.10 – **ROPUST replaces the classifier of already robust models, enhancing their adversarial robustness.** Only the ROPUST classifier needs fine-tuning ; the convolutional stack is frozen. Convolutional features first go through a fully-connected layer, before binarization for use in the Optical Processing Unit (OPU). The OPU performs a non-linear random projection, with *fixed unknown parameters*. A fully-connected layer is then used to obtain a prediction from the output of the OPU. Direct Feedback Alignment is used to train the layer underneath the OPU.

5.6.2 Our defense

ROPUST To enhance the adversarial robustness of pretrained robust models, we propose to replace their classifier with the ROPUST module (Figure 5.10), studied in the first part for networks trained from scratch. We use robust models from the RobustBench model zoo, extracting and freezing their convolutional stack. The robust convolutional features go through a fully connected layer and a binarization step (a sign function), preparing them for the OPU. The OPU then performs a non-linear random projection, with fixed unknown parameters. Lastly, the predictions are obtained through a final fully-connected layer. While the convolutional layers are frozen, we train the ROPUST module on natural data using DFA to bypass the non-differentiable photonic hardware.

Attacking ROPUST. While we could use DFA to attack ROPUST, previous work has shown that methods devoid of weight transport are not effective in generating compelling adversarial examples [Akrout, 2019b]. Therefore, we instead use backward pass differentiable approximation (BPDA) when attacking our defense. For BPDA, we need to find a good differentiable relaxation to non-differentiable layers. For the binarization function, we simply use the derivative of tanh in the backward pass, while we approximate the transpose of the obfuscated parameters with a different fixed random matrix drawn at initialization of the module. More specifically, we consider the expression for the forward pass of the ROPUST module :

$$\boldsymbol{y} = \text{softmax}(\mathbf{W}_3|\mathbf{U}\,\text{sign}(\mathbf{W}_1\mathbf{x})|^2). \tag{5.4}$$

In the backward pass, we substitute $\mathbf{U}^T$ (that we do not have access to) with a different fixed random matrix $\mathbf{R}$, in a setup similar to Feedback Alignment [Lillicrap, 2014]. We also relax the sign function derivative to the derivative of tanh.
We present empirical results on RobustBench in the following Section 5.7. We then ablate the

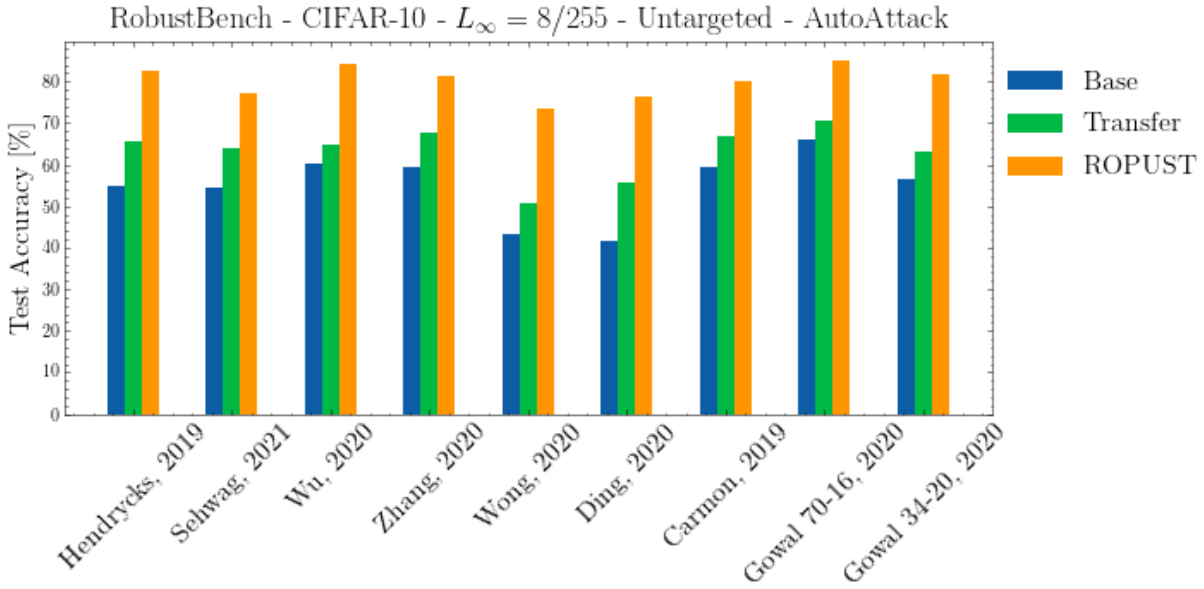


FIGURE 5.11 – **ROPUST systematically improves the test accuracy of already robust models**. Transfer refers to the performance when attacks are generated on the base model and transferred to the ROPUST model. Models from the RobustBench model zoo : Hendrycks et al., 2019 [Hendrycks, 2019], Sehwag et al., 2021 [Sehwag, 2021], Wu et al., 2020 [Wu, 2020], Zhang et al., 2020 [Zhang, 2020], Wong et al., 2020 [Wong, 2020], Ding et al., 2020 [Ding, 2020], Carmon et al., 2019 [Carmon, 2019], Gowal et al., 2020 [Gowal, 2020].

components of our defense in Section 5.8, demonstrating its holistic nature, and we finally create a phase retrieval attack to challenge parameter obfuscation in Section 5.9.

5.7 Evaluating ROPUST on RobustBench

All of the attacks are performed on CIFAR-10 [Krizhevsky, 2009a], using a differentiable backward pass approximation [Athalye, 2018b] as explained in Section 5.6.2. For our experiments, we use OPU input size 512 and output size 8000. We use the Adam optimizer [Kingma, 2014], with learning rate 0.001, for 10 epochs. The process typically takes as little as 10 minutes on a single NVIDIA V100 GPU.

We show our results on nine different models in RobustBench in Figure 5.11. The performance of the original pretrained models from the RobustBench leaderboard is reported as *Base*. *ROPUST* represents the same models equipped with our defense. Finally, *Transfer* indicates the performance of attacks created on the original model and transferred to fool the ROPUST defense. For all models considered, ROPUST improves the robustness significantly, even under transfer.

For transfer, we also tested crafting the attacks on the *Base* model while using the loss of the ROPUST model for the learning rate schedule of APGD. We also tried to use the predictions of ROPUST, instead of the base model, to *remove* the samples that were successfully attacked from the next stage of the ensemble ; however, these modifications did not improve transfer performance. Finally, we remark the robustness increase typically comes at no cost in natural accuracy ; i.e. it is roughly the same for *Base* and *ROPUST* models as shown in Figure 5.12.

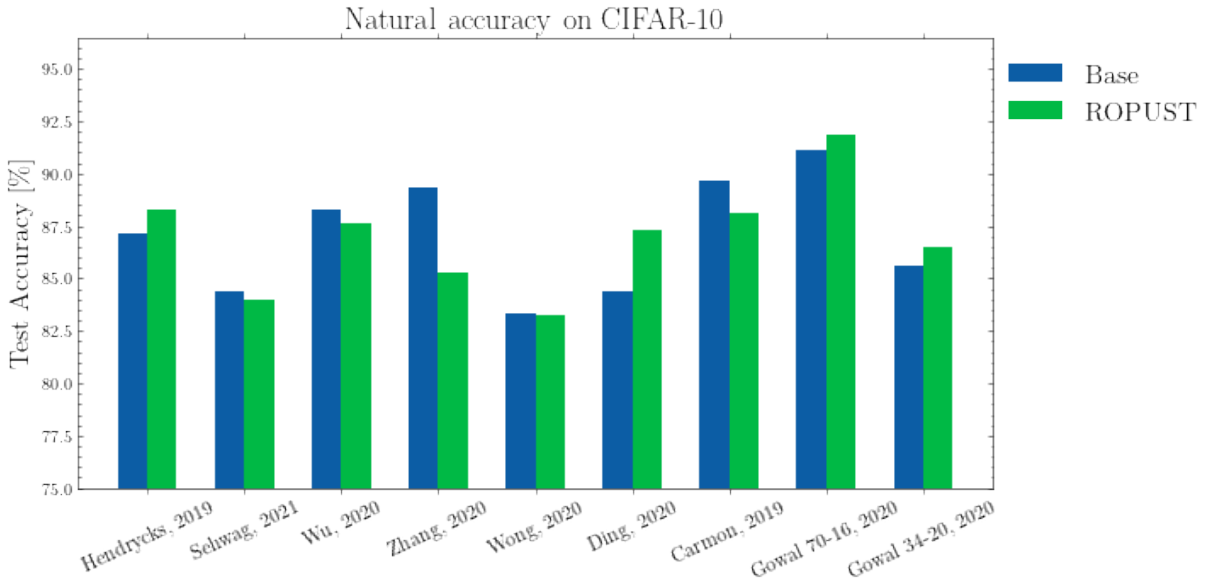


FIGURE 5.12 – **Our ROPUST defense comes at no cost in natural accuracy.** In some cases, natural accuracy is even improved. The model from Zhang, 2020 [Zhang, 2020] is an isolated exception. *Base* models from the RobustBench model zoo : Hendrycks et al., 2019 [Hendrycks, 2019], Sehwag et al., 2021 [Sehwag, 2021], Wu et al., 2020 [Wu, 2020], Zhang et al., 2020 [Zhang, 2020], Wong et al., 2020 [Wong, 2020], Ding et al., 2020 [Ding, 2020], Carmon et al., 2019 [Carmon, 2019], Gowal et al., 2020 [Gowal, 2020].

5.8 Understanding ROPUST : an ablation study

We use the model from [Wong, 2020] available in the RobustBench model zoo to perform our ablation studies. It consists in a PreAct ResNet-18 [He, 2016], pretrained with a *"revisited"* FGSM of increased effectiveness.

Holistic defense. We conduct an ablation study by removing a single component of our defense at a time in simulation : binarization, DFA, parameter obfuscation, and non-linearity $|.|^2$ of the random projection. To remove DFA, we also remove the binarization step and train the ROPUST module with backpropagation, since we have access to the transpose of the transmission matrix in the simulated setting of the ablation study. We show the results in Figure 5.13 : we see that removing the non-linearity $|.|^2$ and the binarization does not have an effect, with the robustness given by *parameter obfuscation* and DFA, as expected. However, note that $|.|^2$ is central to preventing trivial phase retrieval, and is hence a key component of our defense.

Robustness to Square attack While the ablation study on the APGD attack is able to pinpoint the exact sources of robustness for a white-box attack, the same study on the black-box Square attack has surprising results. Indeed, as shown in Figure 5.14, no element of the ROPUST mechanism can be linked to robustness against Square attack. Interestingly, we found an identical behaviour when retraining the standard fully connected classification layer from scratch on natural (non perturbed) data, shown in the same Figure 5.14 under the *Defense-free* label.

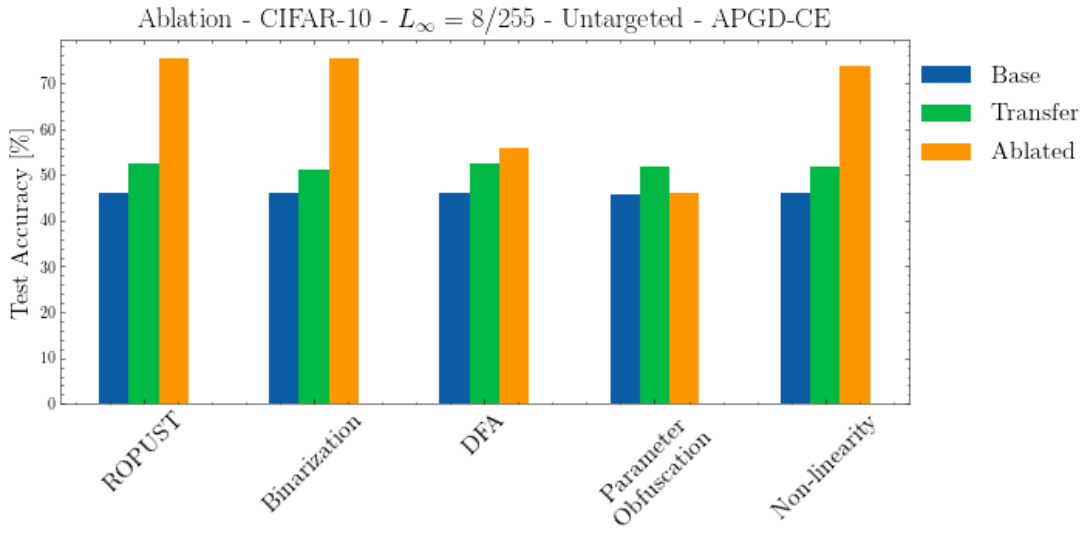


FIGURE 5.13 – **Removing either parameter obfuscation or DFA from our defense causes a large drop in accuracy.** This confirms the intuition that robustness is given by the inability to efficiently generate attacks in a white-box settings when the parameters are obfuscated, and that DFA is capable of generating partially robust features. We note that even though the non-linearity $|.|^2$ does not contribute to robustness, it is key to obfuscation, preventing trivial retrieval. Transfer performance does not change much when removing components of the defense. While the `Base` model is not ablated, we leave its performance as a term of comparison.

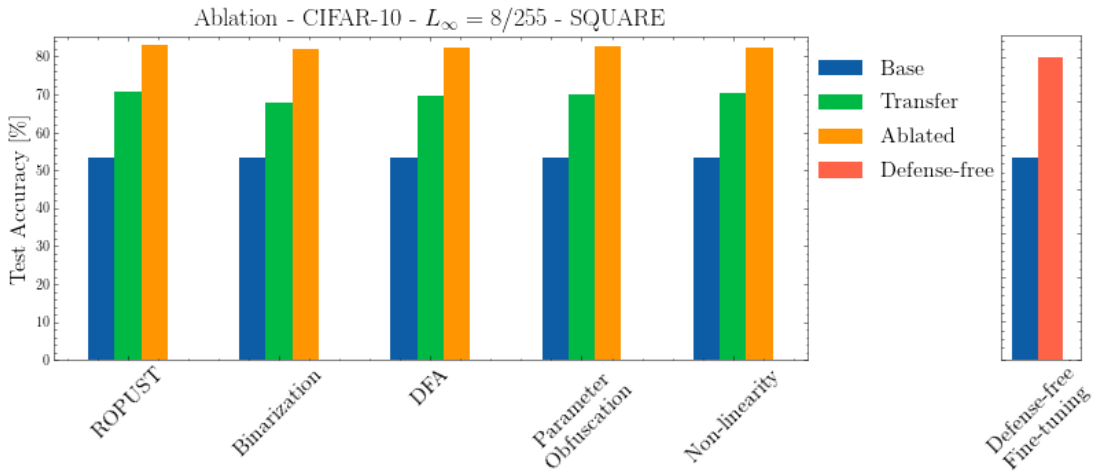


FIGURE 5.14 – **Square attack can be evaded by simply retraining on natural data the classifier of a robust model.** We confirm the same result when retraining the standard fully connected classification layer in the pretrained models in place of the ROPUST module (*Defense-free* result in the chart on the right). While the `Base` model is not ablated, we leave its performance as a term of comparison.

5.9 Phase retrieval attack

Our defense leverages parameter obfuscation to achieve robustness. Yet, however demanding, it is still technically possible to recover the parameters through phase retrieval schemes [Gupta, 2019a ; Gupta, 2020a]. To provide a thorough and fair evaluation of our attack, we study in this section *phase retrieval* attacks. We first consider an idealized setting, and then confront this setting with

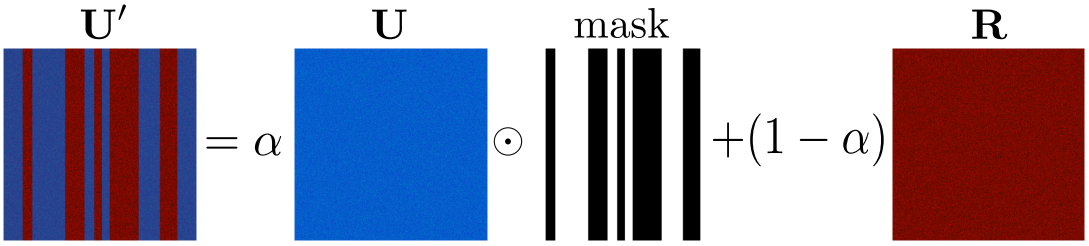


FIGURE 5.15 – **Simplified modelling of phase retrieval.** The retrieved matrix $\mathbf{U}'$ is modeled as the linear interpolation between the real transmission matrix $\mathbf{U}$ and a random matrix $\mathbf{R}$, only for some columns selected by a mask. Varying the value of α and the percentage of masked columns allows to modulate the knowledge of the attacker without running resource-hungry phase retrieval algorithms.

a real-world phase retrieval algorithm from [Gupta, 2020a].

Ideal retrieval model. We build an idealized phase retrieval attack, where the attacker knows a certain fraction of columns, up to a certain precision, schematized in Figure 5.15. To smoothly vary the precision, we model the retrieved matrix $\mathbf{U}'$ as a linear interpolation of the real transmission matrix $\mathbf{U}$ and a completely different random matrix $\mathbf{R}$:

$$\mathbf{U}' = \alpha\mathbf{U} + (1-\alpha)\mathbf{R}. \tag{5.5}$$

In real phase retrieval, this model is valid for a certain fraction of columns of the transmission matrix, and the remaining ones are modeled as independent random vectors. We can model this with a Boolean mask matrix $\mathbf{M}$, so our retrieval model in the end is :

$$\mathbf{U}' = \alpha\mathbf{U} \odot \mathbf{M} + (1-\alpha)\mathbf{R}. \tag{5.6}$$

In this setting, we vary the knowledge of the attacker from the minimum to the maximum by varying α and the percentage of retrieved columns, and we show how the performance of our defense changes in Figure 5.16. In this simplified model only a crude knowledge of the parameters seems sufficient, given the sharp phase transition. We now need to chart where state-of-the-art retrieval methods are on this graph to estimate their ability to break our defense.

Real-world retrieval performance. State-of-the-art phase retrieval methods seek to maximize output correlation, i.e. the correlation on $\boldsymbol{y}$ in Equation 5.1, in place of the correlation with respect to the parameters of the transmission matrix, i.e. $\mathbf{U}$ in Equation 5.1. This leads to a retrieved matrix that may well approximate the OPU outputs, but not the actual transmission matrix it implements. We find this is a significant limitation for attackers. In Figure 5.16, following numerical experiments, we highlight with a white contour the operating region of a state-of-the-art phase retrieval algorithm [Gupta, 2020a], showing that it can manage to only partially reduce the robustness of ROPUST.

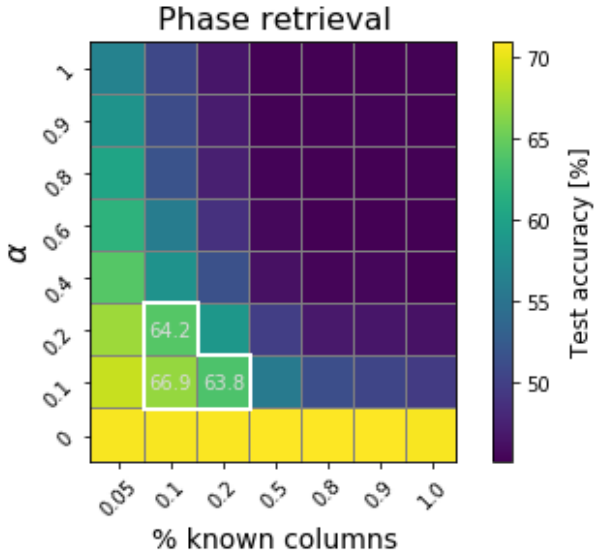


FIGURE 5.16 – **Performance of an APGD-CE attack with a retrieved matrix in place of the, otherwise unknown, transpose of the transmission matrix.** As expected, a better knowledge of the transmission matrix, i.e. higher alpha and/or higher percentage of known columns correlates with the success of the attack, with a sharp phase transition. At first glance, it may seem that even a coarse-grained knowledge of the TM can help the attacker. However, optical phase retrieval works on the output correlation only : accordingly, we find that even state-of-the-art phase retrieval methods operates only in the white contoured region, where the robustness is still greater than the *Base* models. We highlighted the accuracies achieved under attack in this region in the heat-map.

5.10 Conclusion and outlooks

We introduced a new defense technique against white-box, black-box and transfer attacks, based on the implementation of a neural network layer using an optical co-processor. Our method incurs no additional computational cost at training time, and comes at no natural accuracy cost. We evaluated our method in each setting on the CIFAR-10 and CIFAR-100 datasets, against various attacks. In the white-box setting, our defense is robust by design thanks to *parameter obfuscation*. We attempted to adapt white-box attacks to break this defense, testing two different differentiable approximations, however both resulted in less convincing adversarial examples. Furthermore, in the black-box setting, our defense improves robustness by 22% against the strongest black-box attack that we tested. We also showed increased robustness to transfer of attacks, and showed that there exists in this instance a robustness-accuracy trade-off.

To understand robustness to black-box and transfer of attacks, which cannot be explained by obfuscation, we performed an ablation study of our defense. In black-box scenarios, we found that the combined use of a random projection and binarization significantly strengthen our defense, highlighting how the OPU is at the center-stage of our defense. In transfer of attacks, surprisingly, we found that the hybrid training method we devised to train neural networks incorporating our defense builds robust features, enabling our defense layer to be robust against transfer of attacks. These findings motivate further studies on the loss landscape explored by

alternative training methods. In essence, we find that every aspect of our strategy, from the analog optical co-processor and its binarized input, to the hybrid training method we use, contribute in different settings to building an all-around robust defense. In particular, while our defense can be simulated, its physical implementation by an optical co-processor guarantees the parameters remain obfuscated even if the host system is compromised – a unique feature.

Then, we used our defense as a drop-in module to enhance the adversarial robustness of pretrained already robust models, which we named ROPUST. We thoroughly evaluated it on nine different models in the standardized RobustBench benchmark, reaching and improving state-of-the-art performance. In light of these results, we encourage to extend RobustBench to include parameter obfuscation methods.

We performed an ablation study in the white-box setting, confirming our intuition and the results from the first part : the robustness comes from the parameter obfuscation and from the hybrid Backpropagation-DFA training algorithm. The non-linearity $|.|^2$ on the random projection, while not contributing to robustness on its own, is key to prevent trivial deobfuscation by hardening ROPUST against phase retrieval. A similar study in the black-box setting was inconclusive. However, it shed light on a phenomenon of increased robustness against Square attack when retraining from scratch the classifier of robust architectures on natural data. This phenomenon appears to be universal, i.e. independent of the structure of the classification module being fine-tuned, warranting further study.

Finally, we developed a new kind of attack, *phase retrieval attacks*, specifically suited to parameter obfuscation defense such as ours, and we tested their effectiveness. We found that the typical precision regime of even state-of-the-art phase retrieval methods is not enough to completely break ROPUST.

Future work could investigate how the robustness varies with the input and output size of the ROPUST module, and if there are different parameter obfuscation trade-offs when such dimensions change. The combination of ROPUST with other defense techniques, such as adversarial label-smoothing [Goibert, 2019], could also be of interest to further increase robustness. By combining beyond silicon hardware and beyond backpropagation training methods, our work highlights the importance of considering solutions outside of the hardware lottery [Hooker, 2020].

Chapter 6

Photonic Differential Privacy with Direct Feedback Alignment

This chapter is based on [Ohana, 2021].

Optical Processing Units (OPUs) – low-power photonic chips dedicated to large scale random projections – have been used in previous work to train deep neural networks using Direct Feedback Alignment (DFA), an effective alternative to backpropagation. Here, we demonstrate how to leverage the intrinsic noise of optical random projections to build a differentially private DFA mechanism, making OPUs a solution of choice to provide a *private-by-design* training. We provide a theoretical analysis of our adaptive privacy mechanism, carefully measuring how the noise of optical random projections propagates in the process and gives rise to provable Differential Privacy. Finally, we conduct experiments demonstrating the ability of our learning procedure to achieve solid end-task performance.

6.1 Introduction

The widespread use of machine learning models has created concern about their release in the wild when trained on sensitive data such as health records or queries in data bases [Berger, 2019 ; Johnson, 2018]. Such concern has motivated a abundant line of research around privacy-preserving training of models. A popular technique to guarantee privacy is *differential privacy* (DP), that works by injecting noise in an deterministic algorithm, making the contribution of a single data-point hardly distinguishable from the added noise. Therefore it is impossible to infer information on individuals from the aggregate.

While there are alternative methods to ensure privacy, such as knowledge distillation (e.g. PATE [Papernot, 2018]), a simple and effective strategy is to use perturbed and quenched Stochastic Gradient Descent (SGD) [Abadi, 2016] : the gradients are clipped before being aggregated and then perturbed by some additive noise, finally they are used to update the parameters. The DP

property comes at a cost of decreased utility. These biased and perturbed gradients provide a noisy estimate of the update direction and decrease utility, i.e. end-task performance.
We revisit this strategy and develop a private-by-design learning algorithm, inspired by the implementation of an alternative training algorithm, Direct Feedback Alignment [Nøkland, 2016c], on Optical Processing Units [LightOn, 2020], photonic co-processors dedicated to large scale random projections. The analog nature of the photonic co-processor implies the presence of noise, and while this is usually minimized, in this case we propose to leverage it, and tame it to fulfill our needs, *i.e* to control the level of privacy of the learning process. The main sources of noise in Optical Processing Units can be modeled as additive Poisson noise on the output signal, and approach a Gaussian distribution in the operating regime of the device. In particular, these sources can be modulated through temperature control, in order to attain a desired privacy level. Finally, we test our algorithm using the photonic hardware demonstrating solid performance on the goal metrics. To summarize, our setting consists in OPUs performing the multiplication by a fixed random matrix, with a different realization of additive noise for every random projection.

6.1.1 Related work

The amount of noise needed to guarantee differential privacy was first formalized in [Dwork, 2006a]. Later, a training algorithm that satisfied Renyi Differential Privacy was proposed in [Abadi, 2016]. This sparked a line of research in differential privacy for deep learning, investigating different architecture and clipping or noise injection mechanisms [Abay, 2018 ; Ács, 2019]. The majority of these works though rely on backpropagation. An original take was offered in [Lee, 2020], that evaluated the privacy performance of Direct Feedback Alignment (DFA) [Nøkland, 2016c], an alternative to backpropagation. While Lee et al. [Lee, 2020] basically extend the gradient clipping/Gaussian mechanism approach to DFA, our work, while applied to the same DFA setting, is motivated by a photonic implementation that naturally induces noise that we exploit for differential privacy. As such, we provide a new DP framework together with its theoretical analysis.

6.1.2 Motivations and contributions

We propose a hardware-based approach to Differential Privacy (DP), centered around a photonic co-processor, the OPU. We use it to perform optical random projections for a differentially private DFA training algorithm, leveraging noise intrinsic to the hardware to achieve *privacy-by-design*. This is a significant departure from the classical view that such analog noise should be minimized, instead leveraging it as a key feature of our approach. Our mechanism can be formalized through the following (simplified) learning rule at layer ℓ :

$$\delta \mathbf{W}^{\ell} = -\eta \left[(\underbrace{\mathbf{B}^{\ell}\mathbf{e}}_{\text{scaled DFA learning signal}} + \overbrace{\boldsymbol{g}}^{\text{Gaussian hardware noise}}) \odot \phi'_{\ell}(\mathbf{z}^{\ell}) \right] (\underbrace{\mathbf{h}^{\ell-1}}_{\text{neuron-wise clipped activations}})^{\top}. \tag{6.1}$$

Photonic-inspired and tested. The OPU is used as both inspiration and an actual platform

for our experiments. We demonstrate theoretically that the noise induced by the analog co-processor makes the algorithm private by design, and we perform experiments on a real photonic co-processor to show we achieve end-task performance competitive with DFA on GPU.

Differential Privacy beyond backpropagation. We extend previous work [Lee, 2020] on DP with DFA both theoretically and empirically. We provide new theoretical elements for noise injected on the DFA learning signal, a setting closer to our hardware implementation.

Theoretical contribution. Previous works on DP and DFA [Lee, 2020] proposes a straightforward extension of the DP-SGD paradigm to Direct Feedback Alignment. In our work, by adding noise directly to the random projection in DFA, we study a different Gaussian mechanism [Mironov, 2017] with a covariance matrix depending on the values of the activations of the network. Therefore the theoretical analysis is more challenging than in [Lee, 2020]. We succeed to upper bound the Rényi Divergence of our mechanism and deduce the (ε, δ)-DP parameters of our setup.

6.2 Background

Formally the problem we study is the following : the analysis of the built-in Differential Privacy thanks to the combination of DFA and OPUs to train deep architectures. Before proceeding, we recall a minimal set of principles of DFA and Differential Privacy.

From here on $\{\boldsymbol{x}_i\}_{i=1}^N$ are the training points belonging to $\mathbb{R}^d$, $\{y_i\}_{i=1}^N$ the target labels belonging to $\mathbb{R}$. The aim of training a neural network is to find a function $f : \mathbb{R}^d \to \mathbb{R}$ that minimizes the *true* $\mathcal{L}$*-risk* $\mathbb{E}_{XY\sim D}\mathcal{L}(f(X), Y)$, where $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a loss function and D a fixed (and unknown) distribution over data and labels (and the $(\boldsymbol{x}_i, y_i)$ are independent realizations of X, Y), and to achieve that, the *empirical* risk $\frac{1}{N}\sum_{i=1}^N \mathcal{L}(f(\boldsymbol{x}_i), y_i)$ is minimized.

6.2.1 Learning with Direct Feedback Alignment (DFA)

DFA is a biologically inspired alternative to backpropagation with an asymmetric backward pass. For ease of notation, we introduce it for fully connected networks but it generalizes to convolutional networks, transformers and other architectures [Launay, 2020a]. It has been theoretically studied in [Lillicrap, 2016 ; Refinetti, 2021]. Note that in the following, we incorporate the bias terms in the weight matrices.

Forward pass. In a model with L layers of neurons, $\ell \in \{1, \dots, L\}$ is the index of the ℓ-th layer, $\mathbf{W}^\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ the weight matrix between layers $\ell - 1$ and ℓ, ϕ_ℓ the activation function of the neurons, and $\mathbf{h}_\ell$ their activations. The forward pass for a pair $(\boldsymbol{x}, y)$ writes as :

$$\forall \ell \in \{1, \dots, L\} : \mathbf{z}^\ell = \mathbf{W}^\ell \mathbf{h}^{\ell-1}, \mathbf{h}^\ell = \phi_\ell(\mathbf{z}^\ell), \tag{6.2}$$

where $\mathbf{h}^0 \doteq \boldsymbol{x}$ and $\hat{\mathbf{y}} \doteq \mathbf{h}^L = \phi_L(\mathbf{z}^L)$ is the predicted output.

Backpropagation update. With backpropagation [Rumelhart, 1986], leaving aside the specifics of the optimizer used (learning rate, etc.), the weight updates are computed using the chain-rule

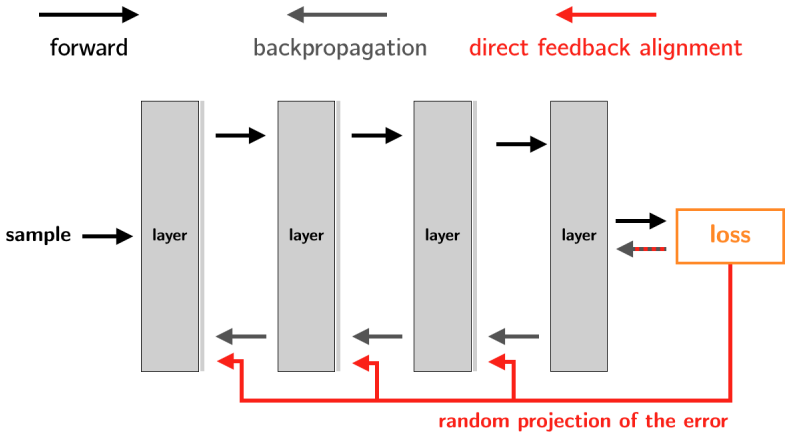


FIGURE 6.1 – Schematic comparison of Backpropagation and Direct Feedback Alignment. The two approaches differ in how the loss impacts each layer of the model. While in backpropagation, the loss is propagated sequentially backwards, in DFA, it directly acts on each layer after random projection.

of derivatives :

$$\delta \mathbf{W}^{\ell} = -\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{\ell}} = -[((\mathbf{W}^{\ell+1})^{\top} \delta \mathbf{z}^{\ell+1}) \odot \phi'_{\ell}(\mathbf{z}^{\ell})](\mathbf{h}^{\ell-1})^{\top}, \delta \mathbf{z}^{\ell} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{\ell}}, \tag{6.3}$$

where ϕ'_{ℓ} is the derivative of ϕ_{ℓ}, $\odot$ is the Hadamard product, and $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ is the prediction loss.

DFA update. DFA replaces the gradient signal $(\mathbf{W}^{\ell+1})^{\top} \delta \mathbf{z}^{\ell+1}$ with a random projection of the derivative of the loss with respect to the pre-activations $\delta \mathbf{z}^{L}$ of the last layer. For losses $\mathcal{L}$ commonly used in classification and regression, such as the squared loss or the cross-entropy loss, this will amount to a random projection of the error $\mathbf{e} \propto \hat{\mathbf{y}} - \mathbf{y}$. With a fixed random matrix $\mathbf{B}^{\ell}$ of appropriate shape drawn at initialization of the learning process, the parameter update of DFA is :

$$\delta \mathbf{W}^{\ell} = -[(\mathbf{B}^{\ell} \mathbf{e}) \odot \phi'_{\ell}(\mathbf{z}^{\ell})](\mathbf{h}^{\ell-1})^{\top}, \mathbf{e} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{L}}. \tag{6.4}$$

Backpropagation vs DFA training. Learning using backpropagation consists in iteratively applying the forward pass Equation 6.2 on batches of training examples and then applying backpropagation updates Equation 6.3. Training with DFA consists in replacing the backpropagation updates by DFA ones Equation 6.4. An interesting feature of DFA is the parallelization of the training step, where all the random projections of the error can be done at the same time, as shown in Figure 6.1. More details about DFA are given in Section 2.3.

6.2.2 Optical Processing Units

An Optical Processing Unit (OPU) [1] is a co-processor that multiplies an input vector $\boldsymbol{x} \in \mathbb{R}^{d}$ by a fixed random matrix $\mathbf{B} \in \mathbb{R}^{D \times d}$, harnessing the physical phenomenon of light scattering

1. Accessible through LightOn Cloud : https://cloud.lighton.ai.

through a diffusive medium [LightOn, 2020]. The operation performed is :

$$\boldsymbol{p} = \mathbf{B}\boldsymbol{x}. \tag{6.5}$$

If the coefficients of $\mathbf{B}$ are unknown, they are guaranteed to be (independently) distributed according to a Gaussian distribution [LightOn, 2020 ; Ohana, 2020]. An additional interesting characteristics of the OPU is its low energy consumption compared to GPUs for high-dimensional matrix multiplication [Ohana, 2020].
A central feature we will rely on is that the measurement of the random projection Equation 6.5 may be corrupted by an additive zero-mean Gaussian random vector $\boldsymbol{g}$, so as for an OPU to provide access to $\boldsymbol{p} = \mathbf{B}\boldsymbol{x} + \boldsymbol{g}$. If $\boldsymbol{g}$ is usually negligible, its variance can be modulated by controlling the physical environment around the OPU. We take advantage of this feature to enforce differential privacy. In the current versions of the OPUs, however, modulating the analog noise at will is not easy, so we will *simulate* the noise numerically in the experiments. More details about the OPU are given in Section 2.4.

6.2.3 Differential Privacy (DP)

Differential Privacy [Dwork, 2006a ; Dwork, 2008] sets a framework to analyze the privacy guarantees of algorithms. It rests on the following core definitions.

Definition 6.2.1 (Neighboring datasets)**.** *Let* $\{\mathcal{X}^j\}_{j=1}^N$ *(e.g.* $\mathcal{X}^j = \mathbb{R}^d$*) be a* domain *and* $\mathcal{D} \doteq \cup_{j=1}^N \mathcal{X}^j$*.* $D, D' \in \mathcal{D}$ *are* neighboring datasets *if they differ from one element. This is denoted by* $\mathcal{D} \sim \mathcal{D}'$*.*

Definition 6.2.2 ((ε, δ)-differential privacy [Dwork, 2008])**.** *Let* $\varepsilon, \delta > 0$*. Let* $\mathcal{A} : \mathcal{D} \to Range\mathcal{A}$ *be a* randomized *algorithm, where* $Range\mathcal{A}$ *is the range of* $\mathcal{D}$ *through* $\mathcal{A}$*.* $\mathcal{A}$ *is* (ε, δ)*-differentially private, or* (ε, δ)*-DP, if for all neighboring datasets* $D, D' \in \mathcal{D}$ *and for all sets* $\mathcal{O} \in Im\ \mathcal{A}$*, the following inequality holds :*

$$\mathbb{P}[\mathcal{A}(D) \in \mathcal{O}] \leq e^{\varepsilon}\mathbb{P}[\mathcal{A}(D') \in \mathcal{O}] + \delta,$$

where the probability relates to the randomness of $\mathcal{A}$*.*

Mironov [Mironov, 2017] proposed an alternative notion of differential privacy based on Rényi α-divergences and established a connection between their definition and the (ε, δ)-differential privacy of Definition 6.2.2. Rényi-based Differential Privacy is captured by the following :

Definition 6.2.3 (Rényi α-divergence [Rényi, 1961])**.** *For two probability distributions* P *and* Q *defined over* $\mathbb{R}$*, the Rényi divergence of order* $\alpha > 1$ *is given by :*

$$\mathbb{D}_\alpha\left(P \| Q\right) \doteq \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim Q}\left(\frac{P(x)}{Q(x)}\right)^\alpha. \tag{6.6}$$

Definition 6.2.4 ((α, ε)-Rényi differential privacy [Mironov, 2017])**.** *Let* $\varepsilon > 0$ *and* $\alpha > 1$*. A randomized algorithm* $\mathcal{A}$ *is* (α, ε)*-Rényi differential private or* (α, ε)*-RDP, if for any neighboring*

datasets $D, D' \in \mathcal{D}$,

$$\mathbb{D}_\alpha \left(\mathcal{A}(D) \| \mathcal{A}(D') \right) \leq \varepsilon.$$

Theorem 6.2.5 (Composition of RDP mechanisms [Mironov, 2017])**.** *Let* $\{M_i\}_{i=1}^k$ *be a set of mechanisms, each satisfying* (α, ε_i)*-RDP. Then their combination is* $(\alpha, \sum_i \varepsilon_i)$*-RDP.*

Going from RDP to the Differential Privacy of Definition 6.2.2 is made possible by the following theorem (see also [Asoodeh, 2020 ; Balle, 2018 ; Wang, 2019]).

Theorem 6.2.6 (Converting RDP parameters to DP parameters [Mironov, 2017])**.** *An* (α, ε)*-RDP mechanism is* $\left(\varepsilon + \frac{\log 1/\delta}{\alpha - 1}, \delta\right)$*-DP for all* $\delta \in (0, 1)$.

For the theoretical analysis, we will need the following proposition, specific to the case of multivariate Gaussian distributions, to obtain bounds on the Rényi divergence.

Proposition 6.2.7 (Rényi divergence for two multivariate Gaussian distributions [Pardo, 2018])**.** *The Rényi divergence for two multivariate Gaussian distributions with means* $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ *and respective covariances* $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ *is given by :*

$$\begin{aligned} \mathbb{D}_\alpha(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \| \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)) = \frac{\alpha}{2} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top (\alpha \boldsymbol{\Sigma}_2 + (1-\alpha)\boldsymbol{\Sigma}_1)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ - \frac{1}{2(\alpha - 1)} \log \left[\frac{\det(\alpha \boldsymbol{\Sigma}_2 + (1-\alpha)\boldsymbol{\Sigma}_1)}{(\det \boldsymbol{\Sigma}_1)^{1-\alpha} (\det \boldsymbol{\Sigma}_2)^\alpha} \right], \end{aligned} \tag{6.7}$$

with $\det(\boldsymbol{\Sigma})$ *the determinant of the matrix* $\boldsymbol{\Sigma}$*. Note that* $(\alpha \boldsymbol{\Sigma}_2 + (1-\alpha)\boldsymbol{\Sigma}_1)^{-1}$ *must be definite-positive*[2]*, otherwise the Rényi divergence is not defined and is equal to* $+\infty$.

Remark 6.2.8. *A standard method to generate an (R)DP algorithm from a deterministic function* $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^d$ *is the* Gaussian mechanism $\mathcal{M}_\sigma$ *acting as* $\mathcal{M}_\sigma \boldsymbol{f}(\cdot) = \boldsymbol{f}(\cdot) + \boldsymbol{v}$ *where* $\boldsymbol{v} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_d)$*. If* $\boldsymbol{f}$ *has* $\Delta_{\boldsymbol{f}}$*- (or* ℓ_2*-)* sensitivity

$$\Delta_{\boldsymbol{f}} \doteq \max_{D \sim D'} \|\boldsymbol{f}(D) - \boldsymbol{f}(D')\|_2,$$

then $\mathcal{M}_\sigma$ *is* $\left(\alpha, \frac{\alpha \Delta_{\boldsymbol{f}}^2}{2\sigma^2}\right)$*-RDP.*

More details about Differential Privacy are given in Section 2.5.2.

6.3 Photonic Differential Privacy

This section explains how to use photonic devices to perform DFA and a theoretical analysis showing how this combination entails *photonic differential privacy.*

2. Note that here $\alpha > 1$ and the combination $\alpha \boldsymbol{\Sigma}_2 + (1-\alpha)\boldsymbol{\Sigma}_1$ is *not* a convex combination ; extra-case must be given to ensure that the resulting matrix is positive

6.3.1 Clipping parameters

As usual in Differential Privacy, we will need to clip layers of the network during the backward pass. Given a vector $\boldsymbol{v} \in \mathbb{R}^d$ and positive constants c, s, ν, we define :

- $\text{clip}_c(\boldsymbol{v}) \doteq (\text{sign}(v_1) \cdot \min(c, |v_1|), \dots, \text{sign}(v_d) \cdot \min(c, |v_d|))^\top$
- $\text{scale}_s(\boldsymbol{v}) \doteq \min(s, \|\boldsymbol{v}\|_2) \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$
- $\text{offset}_\nu(\boldsymbol{v}) \doteq (v_1 + \nu, \dots, v_d + \nu)^\top$

The weight update with clipping to be considered in place of Equation 6.4 is given by

$$\delta \mathbf{W}^\ell = -\frac{1}{m} \sum_{i=1}^{m} (\text{scale}_{s_\ell}(\mathbf{B}^\ell \boldsymbol{e}_i) + \boldsymbol{g}_i) \odot \phi'(\mathbf{z}_i^\ell)) \text{clip}_{c_\ell}(\text{offset}_{\nu_\ell}(\mathbf{h}_i^\ell))^\top. \tag{6.8}$$

For each layer ℓ, we set the s_ℓ, c_ℓ and ν_ℓ parameters as follows :

$$c_\ell \doteq \frac{\tau_h^{max}}{\sqrt{n_\ell}} \qquad \nu_\ell \doteq \frac{\tau_h^{min}}{\sqrt{n_\ell}} \qquad s_\ell \doteq \tau_B. \tag{6.9}$$

These choices ensure that :

- $\tau_h^{min} \leq \|\text{clip}_{c_\ell}(\text{offset}_{\nu_\ell}(\mathbf{h}_i^\ell))\|_2 \leq \tau_h^{max}$
- $\|\text{scale}_{s_\ell}(\mathbf{B}^\ell \boldsymbol{e}_i)\|_2 \leq \tau_B$
- Moreover, we require the derivatives of the each activation function ϕ_ℓ are lower and upper bounded by constants i.e. $\gamma_\ell^{min} \leq |\phi'_\ell(t)| \leq \gamma_\ell^{max}$ for all scalars t. This is a reasonable assumption for activation functions such as sigmoid, tanh, ReLU...

In the following, the quantities should all be considered clipped/scaled/offset as above and, for sake of clarity, we drop the explicit mentions of these operations.

6.3.2 Photonic Direct Feedback Alignment is a natural Gaussian mechanism

Noise modulation. Mainly due to the photon shot noise of the measurement process [Konnik, 2014], Gaussian noise is naturally added to the random projection of Equation 6.5. This noise is negligible for machine learning purposes, however it can be modulated through temperature control yielding the following projection :

$$\boldsymbol{p} = \mathbf{B}\boldsymbol{x} + \mathcal{N}(0, \sigma^2 \mathbf{I}_D), \tag{6.10}$$

where $\mathbf{I}_D$ is the identity matrix in dimension D. Note that this noise is *truly random* due to the quantum nature of the photon shot noise. As previously stated, due to experimental constraints, the noise will be simulated in the experiments of Section 6.4.

Building on that feature, we perform the random projection of DFA of Equation 6.4 using the OPU. Since this equation is valid for any layer ℓ of the network, we allow ourselves, for sake of clarity, to drop the layer indices and study the generic update (the quantities below are all

Algorithm 3 Photonic DFA training

Require: training set $\mathcal{S} = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^{N}$, ϕ_ℓ with bounded derivatives, scale parameters s_ℓ, clipping thresholds ν_ℓ and c_ℓ, stepsize η, noise scale σ, minibatch of size m, number of iterations T.

Ensure: A performing DP model.

for $\ell = 1$ to L **do**
  Sample $\mathbf{B}^\ell$ ▷ Note : with OPUs, there is no explicit sampling of $\mathbf{B}$
end for
for T iterations **do**
  Create a minibatch $S \subset \{1, \dots, N\}$ of size $|S| = m$ (sampling without replacement)
  for $i \in S$ **do**
    for $\ell = 1$ to $L-1$ **do**
      $\mathbf{z}_i^\ell = \mathbf{W}^\ell \mathbf{h}_i^{\ell-1}$
      $\mathbf{h}_i^\ell = \phi_\ell(\mathbf{z}_i^\ell)$
    end for
    $\hat{\boldsymbol{y}}_i \leftarrow \phi_\ell(\mathbf{W}^\ell \mathbf{h}_i^{\ell-1})$
  end for
  for $\ell = L$ to 1 **do**
    Perform $\mathbf{B}^\ell \boldsymbol{e}_i$ with the OPU
    Independently sample $\boldsymbol{g}_1^\ell, \dots, \boldsymbol{g}_m^\ell \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_{n_\ell})$
    $\mathbf{W}^\ell \leftarrow \mathbf{W}^\ell - \frac{\eta}{m} \sum_{i=1}^{m} ((\text{scale}_{s_\ell}(\mathbf{B}^\ell \boldsymbol{e}_i) + \boldsymbol{g}_i^\ell) \odot \phi_\ell'(\mathbf{z}_i^\ell)) \text{clip}_{c_\ell}(\text{offset}_{\nu_\ell}(\mathbf{h}_i^\ell))^\top$
  end for
end for

clipped as in Section 6.3.1) :

$$\delta \mathbf{W} = -\frac{1}{m} \sum_{i=1}^{m} (\mathbf{B}\boldsymbol{e}_i + \boldsymbol{g}_i) \odot \phi'(\mathbf{z}_i)) \mathbf{h}_i^\top \quad \text{(clipped quantities as in section 6.3.1)} \tag{6.11}$$

$$= -\frac{1}{m} \sum_{i=1}^{m} (\mathbf{B}\boldsymbol{e}_i \odot \phi'(\mathbf{z}_i)) \mathbf{h}_i^\top + \frac{1}{m} \sum_{i=1}^{m} (\boldsymbol{g}_i \odot \phi'(\mathbf{z}_i)) \mathbf{h}_i^\top, \tag{6.12}$$

where $\boldsymbol{g}_i \sim \mathcal{N}(0, \sigma^2 I_{n_\ell})$ is the Gaussian noise added during the OPU process. As stated previously, its variance σ^2 can be modulated to obtain the desired value. The overall training procedure with Photonic DFA (PDFA) is described in Algorithm 3.

6.3.3 Theoretical analysis of our method

In the following, the quantities in the DFA update of the weights are always clipped according to Equation 6.11 and as before clipping/scale/offset operators are in force but dropped from the text.

To demonstrate that our mechanism is Differentially Private, we will use the following reasoning : the noise being added at the random projection level as in Equation 6.10, we can decompose the update of the weights as a Gaussian mechanism as in Equation 6.12. We will compute the covariance matrix of the Gaussian noise, which will depend on the data, which is in striking contrast with the standard Gaussian mechanism [Abadi, 2016]. We will then use Proposition 6.2.7 to compute the upper bound the Rényi divergence. The Differential Privacy parameters

will be obtained using Theorem 6.2.6.
In the following, we will consider the Gaussian mechanism applied to the columns of the weight matrix. We consider this case for the following reasons : since our noise matrix has the same realisation of the Gaussian noise (but multiplied by different scalars), it makes sense to consider the Differential Privacy parameters of only columns of the weight matrix and then multiply the Rényi divergence by the number of columns. If our noise was i.i.d. we could have used the theorems from [Chanyaswad, 2018] to lower the Rényi divergence. Given the update equation of the weights at layer ℓ in Equation 6.12, the update of column k of the weight of layer ℓ is the following Gaussian mechanism :

$$\frac{1}{m}\sum_{i=1}^{m}((\mathbf{B}\boldsymbol{e}_i)\odot\phi'(\mathbf{z}_i))h_{ik}+\frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{g}_i\odot\phi'(\mathbf{z}_i))h_{ik}=\boldsymbol{f}_k(D)+\mathcal{N}(0,\boldsymbol{\Sigma}_k), \tag{6.13}$$

where $\boldsymbol{\Sigma}_k = \frac{\sigma^2}{m^2}\mathbf{diag}(\boldsymbol{a}_k)^2$ and $(\boldsymbol{a}_k)_j = \sqrt{\sum_{i=1}^{m}(\phi'_{ij}h_{ik})^2}, \forall j = 1,\dots,n_{\ell-1}$. Note that these expressions are due to the inner product with $\mathbf{h}_i$. In the following, we will focus on column k and we therefore drop the index k in the notation. Using the clipping of the quantities of interest detailed in Equation 6.8, we can compute some useful bounds bounds on a_j :

$$\sqrt{\frac{m}{n_\ell}}\gamma_\ell^{min}\tau_h^{min}\leq a_j\leq\sqrt{\frac{m}{n_\ell}}\gamma_\ell^{max}\tau_h^{max}. \tag{6.14}$$

Proposition 6.3.1 (Sensitivity of Photonic DFA [Lee, 2020])**.** *For neighboring datasets D and D' (i.e. differing from only one element), the sensitivity Δ_f^ℓ of the function $\boldsymbol{f}_k$ described in Equation 6.12 at layer ℓ is given by :*

$$\Delta_{\boldsymbol{f}}^\ell = \sup_{D\sim D'}\|\boldsymbol{f}(D)-\boldsymbol{f}(D')\|_2\leq\frac{2}{m}\|(\mathbf{B}^\ell\boldsymbol{e}_i)\odot\phi'_\ell(\mathbf{z}_i^\ell))h_{ik}^{\ell-1}\|_2 \tag{6.15}$$

$$\leq\frac{2}{m}\tau_B\gamma_\ell^{max}\frac{\tau_h^{max}}{\sqrt{n_\ell}}. \tag{6.16}$$

The following proposition is our main theoretical result : we compute the ε parameter of Rényi Differential Privacy.

Proposition 6.3.2 (Photonic Differential Privacy parameters)**.** *Given two probability distributions $P\sim\mathcal{N}(\boldsymbol{f}(D),\boldsymbol{\Sigma})$ and $Q\sim\mathcal{N}(\boldsymbol{f}(D'),\boldsymbol{\Sigma}')$ corresponding to the Gaussian mechanisms depicted in Equation 6.13 on neighboring datasets D and D', the Rényi divergence of order α between these mechanisms is :*

$$\mathbb{D}_\alpha(P\|Q)\leq\frac{2\alpha}{m.\sigma^2}\frac{(\gamma^{max}\tau^{max}\tau_B)^2}{(\gamma_\ell^{min}\tau_h^{min})^2}+\frac{n_\ell.\alpha}{2(\alpha-1)}\log\left[\frac{m(\gamma_\ell^{min}\tau_h^{min})^2}{(m+1)(\gamma_\ell^{min}\tau_h^{min})^2-(\gamma_\ell^{max}\tau_h^{max})^2}\right]$$

$$=\varepsilon_{PDFA}. \tag{6.17}$$

Our mechanism is therefore $(\alpha,T\varepsilon_{PDFA})$-RDP with T the number of training epochs. We can deduce that the mechanism on the weight matrix with $n_{\ell-1}$ columns is $(\alpha,T\varepsilon_{PDFA})$-RDP. Then the mechanism of the whole network composed of L layers is $(\alpha,LT\varepsilon_{PDFA})$-RDP. We can then convert our bound to DP parameters using Theorem 6.2.6 to obtain a $(LT\varepsilon_{PDFA}+\frac{\log 1/\delta}{\alpha-1},\delta)$-DP

mechanism for all $\delta \in (0, 1)$.

Proof. In the following, the variables with a prime correspond to the ones built upon dataset D'. According to Equation 6.13, the covariance matrices $\mathbf{\Sigma}$ and $\mathbf{\Sigma}'$ are diagonal and any of their weighted sum is diagonal, as well as their inverse. Moreover, the determinant of a diagonal matrix is the product of its diagonal elements. Using these elements in Equation 6.7 yields :

$$\mathbb{D}_\alpha(P\|Q) = \sum_{j=1}^{n_\ell} \left(\frac{\alpha m^2}{2\sigma^2} \frac{(f_j(D) - f_j(D'))^2}{\alpha\, a_j'^2 + (1-\alpha)a_j^2} - \frac{1}{2(\alpha-1)} \log \left[\frac{(1-\alpha)a_j^2 + \alpha a_j'^2}{a_j^{2(1-\alpha)} a_j'^{2\alpha}} \right] \right).$$

Using the fact that we are studying neighboring datasets, the sums composing a_j and a'_j differ by only one element at element $i = I$. This implies that

$$\alpha\, a_j'^2 + (1-\alpha)a_j^2 = a_j^2 + \alpha[(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2 - (\phi'_{Ij}h_{Ik})^2)].$$

where $\tilde{\phi}'_{Ij}$ and $\tilde{h}^2_{Ik}$ are taken on the dataset D'. By choosing D and D' such that $[(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2 - (\phi'_{Ij}h_{Ik})^2)] \geq 0$ and some rearrangement, we can upper bound the Rényi divergence by :

$$\mathbb{D}_\alpha(P\|Q) \leq \frac{\alpha.m^2}{2\sigma^2} \frac{\Delta_{\boldsymbol{f}}^2}{(\sqrt{\frac{m}{n_\ell}}\gamma_\ell^{min}\tau_h^{min})^2} + \frac{\alpha.n_\ell}{2(\alpha-1)} \log \left[\frac{m(\gamma_\ell^{min}\tau_h^{min})^2}{(m+1)(\gamma_\ell^{min}\tau_h^{min})^2 - (\gamma_\ell^{max}\tau_h^{max})^2} \right].$$

Using the bound on the sensitivity $\boldsymbol{f}$ computed in Equation 6.15, we obtain the desired $\varepsilon_{\text{PDFA}}$, upper bound of the Rényi divergence. A more detailed proof is presented in Appendix 6.6.1. □

Remark 6.3.3. *We started by computing the RDP parameters on a column of the weight matrix linking layers* $\ell - 1$ *to* ℓ *then generalized it to the whole matrix. The passage from the vectorial to the matrix case yields the same* ε *in the RDP mechanism, as concatenation is not a composition mechanism. We can make the following analogy : the value of the Rényi divergence between two Gaussian random variables with variance* σ^2 *is the same as the one between two Gaussian vectors of Covariance matrix* $\sigma^2\mathbf{I}$ *(with suitable means).*

Remark 6.3.4. *This bound is not tight since it assumes that all the activations reach their worst cases in all the layers for upper bounding. However, obtaining a tighter bound would be very challenging since the values of the covariance matrices depend on the output of the neurons of the Neural network, which are data and architecture dependent.*
We believe tighter bounds could be obtained in much simpler cases. First we can notice that having equal covariance matrices $\mathbf{\Sigma}$ *and* $\mathbf{\Sigma}'$ *would cancel the logarithm term. If additionally we assume that all the activations saturate to their clipping values, then we would retrieve the formula of* ε *in [Lee, 2020].*
Owing to mini-batch training, we believe the privacy parameter could be further improve by considering subsampling mechanism [Wang, 2019] and its properties. However, this would require a novel theoretical framework adapted to our case and we leave it for future work.

6.4 Experimental results

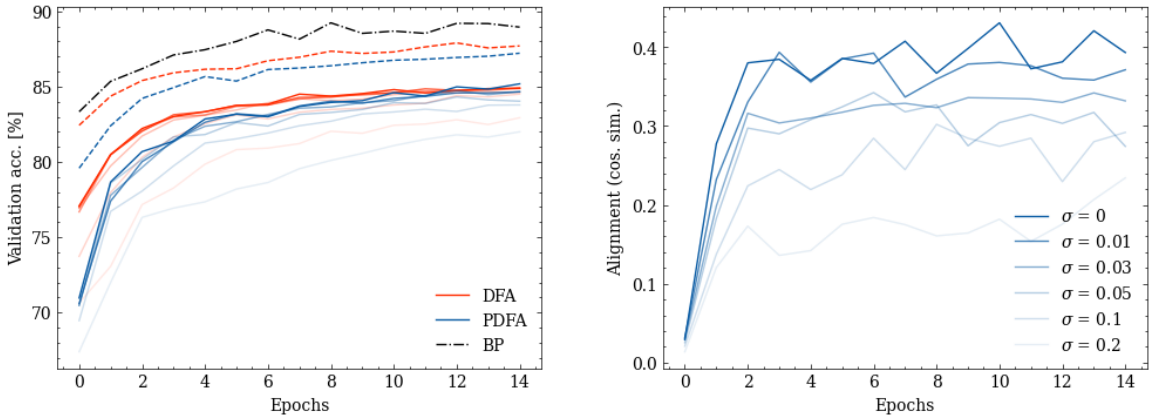


FIGURE 6.2 – **Photonic training on Fashion-MNIST.** Left : BP, DFA, and photonic DFA (PDFA) training runs for various degrees of privacy. Dashed runs (**- -**) are non-private. Increasingly transparent runs have increased noise, see Table 6.1 for details. PDFA is always very close to DFA performance, and both are robust to noise. Right : gradient alignment (cosine similarity between PDFA and BP gradients) for the second layer of the network, at varying degrees of noise. Increasing noise degrades alignment, but alignment values remain high enough to support learning.

In this section, we demonstrate that photonic training is robust to Gaussian mechanism, i.e. adding noise as in Equation 6.8, delivering good end-task performance even under strong privacy constraints. As detailed by our theoretical analysis, we focus on two specific mechanisms :

- **Clipping and offsetting the neurons** with $\frac{\tau_h^{max}}{\sqrt{n_\ell}}$ and $\frac{\tau_h^{min}}{\sqrt{n_\ell}}$ to enforce $\tau_h^{min} \leq \|\mathbf{h}^\ell\|_2 \leq \tau_h^{max}$, as explained in section 6.3.1 ;
- **Adding noise g to Be**, according to the clipping of **Be** with τ_B ($||\mathbf{Be}||_2 \leq \tau_B$) and the scaling of **g** with σ ($\mathbf{g} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$).

To make our results easy to interpret, we fix $\tau_B = 1$, such that $\sigma = 0.1$ implies $||\mathbf{g}||_2 \simeq ||\mathbf{Be}||_2$. At $\sigma = 0.01$, this means that the noise is roughly 10% of the norm of the DFA learning signal. This is in line with differentially-private setup using backpropagation, and is in fact more demanding as our experimental setup makes it such that this is a lower bound on the noise.

Photonic DFA. We perform the random projection **Be** at the core of the DFA training step using the OPU, a photonic co-processor. As the inputs of the OPU are binary, we ternarize the error – although binarized DFA, known as Direct Random Target Propagation [Frenkel, 2021], is possible, its performance is inferior. Ternarization is performed using a tunable threshold t, such that values smaller than $-t$ are set to -1, values larger than t are set to 1, and all in between values are set to 0. We then project the positive part $\mathbf{e}_+$ of this vector using the OPU, obtaining $\mathbf{Be}_+$, and then the negative part $\mathbf{e}_-$, obtaining $\mathbf{Be}_-$. Finally, we substract the two to obtain the projection of the ternarized error, $\mathbf{B}(\mathbf{e}_+ - \mathbf{e}_-)$. This is in line with the setup proposed in [Launay, 2020b]. We refer thereafter to DFA performed on a ternarized error on a GPU as

ternarized DFA (TDFA), and to DFA performed optically with a ternarized error as photonic DFA (PDFA).

Setting. We run our simulations on cloud servers with a single NVIDIA V100 GPU and an OPU, for a total estimate of 75 GPU-hours. We perform our experiments on Fashion-MNIST dataset [Xiao, 2017], reserving 10% of the data as validation, and reporting test accuracy on a held-out set. We use a fully-connected network, with two hidden layers of size 512, with tanh activation. Optimization is done over 15 epochs with SGD, using a batch size of 256, learning rate of 0.01 and 0.9 momentum. For TDFA, we use a threshold of 0.15. Despite the fundamentally different hardware platform, no specific hyperparameter tuning is necessary for the photonic training : this demonstrates the reliability and robustness of our approach.

BP baseline. We also apply our DP mechanism to a network trained with backpropagation. The clipping and offsetting of the activations' neurons is unchanged, but we adapt the noise element. We apply the noise on the derivative of the loss once at the top of the network. We also lower the learning rate to 10^{-4} to stabilize runs.

σ	**non-private**	**0**	**0.01**	**0.03**	**0.05**	**0.1**	**0.2**
τ_B				**1**			
BP	88.33	75.22	70.71	71.47	71.27	70.28	66.78
DFA	86.80	84.20	84.04	84.15	83.70	83.06	81.66
TDFA	86.63	84.20	84.38	84.04	83.94	82.98	80.80
PDFA	85.85	84.00	83.79	83.69	83.36	82.63	80.94

TABLE 6.1 – **Test accuracy on Fashion-MNIST with our DP mechanism.** We find our approach to be robust to increasing DP noise σ. In particular, photonic DFA results (PDFA) are always within 1% of the corresponding DFA run.

Results. We fix $\tau_h^{\max} = 1$ for all experiments, and consider a range of σ corresponding to noise between 0-200% of the DFA training signal $\mathbf{Be}$. We also compare to a non-private, vanilla baseline. Results are reported in Table 6.1 and Figure 6.2.
We find our DFA-based approach to be remarkably robust to the addition of noise, providing Differential Privacy, with a test accuracy hit contained within 3% of baseline for up to $\sigma = 0.05$ (i.e. noise 50% as large as the training signal). Most of the performance hit can actually be attributed to the aggressive activation clipping, with noise having a limited effect. In comparison, BP is far more sensitive to activation clipping and to our noise mechanism. However, our method was devised for DFA and not BP, explaining the under-performance of BP. Finally, photonic training achieves good test accuracy, always within 1% of the corresponding DFA run. This demonstrates the validity of our approach, on a real photonic co-processor. We note that, usually, demonstrations of neural networks with beyond silicon hardware are mostly limited to simulations [Hughes, 2018 ; Guo, 2019], or that these demonstrations come with a significant end-task performance penalty [Xu, 2021 ; Wetzstein, 2020]. **Additional results** with similar conclusions on MNIST and CIFAR-10 are presented in Appendix 6.7.

6.5 Conclusion and outlooks

We have investigated how the Gaussian measurement noise that goes with the use of the photonic chips known as Optical Processor Units, can be taken advantage of to ensure a Differentially Private Direct Feedback Alignment training algorithm for deep architectures. We theoretically establish the features of the so-obtained *Photonic Differential Privacy* and we feature these theoretical findings with compelling empirical results showing how adding noise does not decreases the performance significantly.

At an age where both privacy-preserving training algorithms and energy-aware machine learning procedures are a must, our contribution addresses both points through our photonic differential privacy framework. As such we believe the holistic machine learning contribution we bring will mostly bring positive impacts by reducing energy consumption when learning from large-scale datasets and by keeping those datasets private. On the negative impact side, our DP approach is heavily based on clipping, which is well-known to have negative effects on underrepresented classes and groups [Bagdasaryan, 2019 ; Hooker, 2021] in a machine learning model.

We plan to extend the present work in two ways. First, we would like to refine the theoretical analysis and exhibit privacy properties that are more in line with the observed privacy ; this would give us theoretical grounds to help us set parameters such as the clipping thresholds or the noise modulation. Second, we want to expand our training scenario and address wider ranges of applications such as recommendation, federated learning, natural language processing. We also plan to spend efforts so as to mitigate the effect of clipping on the fairness of our model [Xu, 2020].

6.6 Appendix : complete proofs of Differential Privacy parameters

6.6.1 Extended proof of Proposition 6.3.2

As a reminder, we would like to compute the Rényi divergence of the following Gaussian mechanism, where all the quantities are clipped as in Equation 6.1 :

$$\frac{1}{m}\sum_{i=1}^{m}((\mathbf{B}\boldsymbol{e}_i)\odot\phi'(\mathbf{z}_i))h_{ik}+\frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{g}_i\odot\phi'(\mathbf{z}_i))h_{ik}=\boldsymbol{f}_k(D)+\mathcal{N}(0,\boldsymbol{\Sigma}_k), \tag{6.18}$$

where $\boldsymbol{\Sigma}_k=\frac{\sigma^2}{m^2}\mathbf{diag}(\boldsymbol{a}_k)^2$ and $(\boldsymbol{a}_k)_j=\sqrt{\sum_{i=1}^{m}(\phi'_{ij}h_{ik})^2},\forall j=1,\dots,n_{\ell-1}$. As explained in the main text, we will focus on column k and will drop the k indices. The proposition we want to prove is the following :

Proposition 6.3.2 (Photonic Differential Privacy parameters)**.** *Given two probability distributions $P\sim\mathcal{N}(\boldsymbol{f}(D),\boldsymbol{\Sigma})$ and $Q\sim\mathcal{N}(\boldsymbol{f}(D'),\boldsymbol{\Sigma}')$ corresponding to the Gaussian mechanisms depicted in Equation 6.18 on neighboring datasets D and D', the Rényi divergence of order α*

between these mechanisms is :

$$\mathbb{D}_\alpha(P\|Q) \leq \frac{2.\alpha}{m.\sigma^2}\frac{(\gamma^{max}\tau^{max}\tau_B)^2}{(\gamma_\ell^{min}\tau_h^{min})^2} + \frac{n_\ell.\alpha}{2(\alpha-1)}\log\left[\frac{m(\gamma_\ell^{min}\tau_h^{min})^2}{(m+1)(\gamma_\ell^{min}\tau_h^{min})^2 - (\gamma_\ell^{max}\tau_h^{max})^2}\right]$$
$$= \varepsilon_{PDFA}. \tag{6.19}$$

Our mechanism is therefore $(\alpha, T\varepsilon_{PDFA})$*-RDP with* T *the number of training epochs. We can deduce that the mechanism on the weight matrix with* $n_{\ell-1}$ *columns is* $(\alpha, Tn_{\ell-1}\varepsilon_{PDFA})$*-RDP. Then the mechanism of the whole network composed of* L *layers is* $(\alpha, LTn_{\ell-1}\varepsilon_{PDFA})$*-RDP. We can then convert our bound to DP parameters using Theorem 2 to obtain a* $(LTn_{\ell-1}\varepsilon_{PDFA} + \frac{\log 1/\delta}{\alpha-1}, \delta)$*-DP mechanism for all* $\delta \in (0,1)$*.*

Proof. In the following, the variables with a prime correspond to the ones built upon dataset D'. According to Equation 6.18, the covariance matrices $\mathbf{\Sigma}$ and $\mathbf{\Sigma}'$ are diagonal and any of their weighted sum is diagonal, as well as their inverse. Moreover, the determinant of a diagonal matrix is the product of its diagonal elements. Using this in Equation 6.7 yields :

$$\mathbb{D}_\alpha(P\|Q) = \sum_{j=1}^{n_\ell}\left(\frac{\alpha m^2}{2\sigma^2}\frac{(f_j(D) - f_j(D'))^2}{\alpha\, a_j'^2 + (1-\alpha)a_j^2} - \frac{1}{2(\alpha-1)}\log\left[\frac{(1-\alpha)a_j^2 + \alpha a_j'^2}{a_j^{2(1-\alpha)}a_j'^{2\alpha}}\right]\right).$$

Using the fact that we are studying neighboring datasets, the sums composing a_j and a'_j differ by only one element at element $i = I$. This implies that

$$\begin{aligned}\alpha\, a_j'^2 + (1-\alpha)a_j^2 &= \alpha.\sum_{i=1}^{m}(\tilde{\phi}'_{ij}\tilde{h}_{ik})^2 + (1-\alpha).\sum_{i=1}^{m}(\phi'_{ij}h_{ik})^2 \\ &= \sum_{i=1}^{m}(\phi'_{ij}h_{ik})^2 + \alpha.(\sum_{i=1}^{m}(\tilde{\phi}'_{ij}\tilde{h}_{ik})^2 - \sum_{i=1}^{m}(\phi'_{ij}h_{ik})^2) \\ &= a_j^2 + \alpha[(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2 - (\phi'_{Ij}h_{Ik})^2)],\end{aligned}$$

where $\tilde{\phi}'_{Ij}$ and $\tilde{h}^2_{Ik}$ are taken on dataset D'. Inserting this in the Rényi divergence yields :

$$\mathbb{D}_\alpha(P\|Q) = \sum_{j=1}^{n_\ell}\left(\frac{\alpha m^2}{2\sigma^2}\frac{(f_j(D) - f_j(D'))^2}{a_j^2 + \alpha[(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2 - (\phi'_{Ij}h_{Ik})^2)]} - \frac{1}{2(\alpha-1)}\log\left[\frac{a_j^2 + \alpha[(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2 - (\phi'_{Ij}h_{Ik})^2)]}{a_j^{2(1-\alpha)}a_j'^{2\alpha}}\right]\right).$$

By choosing D and D' such that $[(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2 - (\phi'_{Ij}h_{Ik})^2)] \geq 0$, the Rényi divergence is upper bounded as follow :

$$\mathbb{D}_\alpha(P\|Q) \leq \sum_{j=1}^{n_\ell}\left(\frac{\alpha m^2}{2\sigma^2}\frac{(f_j(D) - f_j(D'))^2}{a_j^2} - \frac{1}{2(\alpha-1)}\log\left[\frac{a_j^{2\alpha}}{a_j'^{2\alpha}}\right]\right).$$

Noting that $a_j^2 = \sum_{i=1}^m (\phi'_{ij}h_{ik})^2 + (\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2 - (\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2 = a_j'^2 - [(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2 - (\phi'_{Ij}h_{Ik})^2]$ yields :

$$\begin{aligned}
\mathbb{D}_\alpha(P\|Q) &\leq \sum_{j=1}^{n_\ell}\left(\frac{\alpha m^2}{2\sigma^2}\frac{(f_j(D)-f_j(D'))^2}{a_j^2} - \frac{\alpha}{2(\alpha-1)}\log\left[\frac{a_j^2}{a_j'^2}\right]\right)\\
&\leq \sum_{j=1}^{n_\ell}\left(\frac{\alpha m^2}{2\sigma^2}\frac{(f_j(D)-f_j(D'))^2}{a_j^2} - \frac{\alpha}{2(\alpha-1)}\log\left[\frac{a_j'^2-[(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2-(\phi'_{Ij}h_{Ik})^2]}{a_j'^2}\right]\right)\\
&\leq \sum_{j=1}^{n_\ell}\left(\frac{\alpha m^2}{2\sigma^2}\frac{(f_j(D)-f_j(D'))^2}{a_j^2} + \frac{\alpha}{2(\alpha-1)}\log\left[\frac{a_j'^2}{a_j'^2-[(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2-(\phi'_{Ij}h_{Ik})^2]}\right]\right)\\
&\leq \frac{\alpha m^2}{2\sigma^2}\frac{n_\ell.\Delta_{\boldsymbol{f}}^2}{m(\gamma_\ell^{min}\tau_h^{min})^2} + \sum_{j=1}^{n_\ell}\frac{\alpha}{2(\alpha-1)}\log\left[\frac{a_j'^2}{a_j'^2-[(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2-(\phi'_{Ij}h_{Ik})^2]}\right]\\
&\leq \frac{2.\alpha}{m.\sigma^2}\frac{(\gamma^{max}\tau^{max}\tau_B)^2}{(\gamma_\ell^{min}\tau_h^{min})^2} + \frac{n_\ell.\alpha}{2(\alpha-1)}\log\left[\frac{m(\gamma_\ell^{min}\tau_h^{min})^2}{(m+1)(\gamma_\ell^{min}\tau_h^{min})^2-(\gamma_\ell^{max}\tau_h^{max})^2}\right]\\
&= \varepsilon_{\text{PDFA}},
\end{aligned}$$

where we used the upper bounds on the sensitivity $\Delta_{\boldsymbol{f}}^2$ and $a_j'^2$. This is the result of Proposition 3. □

Note that an alternative expression is :

$$\begin{aligned}
\mathbb{D}_\alpha(P\|Q) &\leq \frac{\alpha m}{2\sigma^2}\frac{n_\ell.\Delta_{\boldsymbol{f}}^2}{(\gamma_\ell^{min}\tau_h^{min})^2} + \sum_{j=1}^{n_\ell}\frac{\alpha}{2(\alpha-1)}\log\left[\frac{a_j'^2}{a_j'^2-[(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2-(\phi'_{Ij}h_{Ik})^2]}\right]\\
&= \frac{\alpha m}{2\sigma^2}\frac{n_\ell.\Delta_{\boldsymbol{f}}^2}{(\gamma_\ell^{min}\tau_h^{min})^2} + \sum_{j=1}^{n_\ell}\frac{\alpha}{2(\alpha-1)}\log\left[\frac{\sum_{i=1}^m(\tilde{\phi}'_{ij}\tilde{h}_{ik})^2}{\sum_{i=1}^m(\tilde{\phi}'_{ij}\tilde{h}_{ik})^2-[(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2-(\phi'_{Ij}h_{Ik})^2]}\right]\\
&= \frac{\alpha m}{2\sigma^2}\frac{n_\ell.\Delta_{\boldsymbol{f}}^2}{(\gamma_\ell^{min}\tau_h^{min})^2} + \sum_{j=1}^{n_\ell}\frac{\alpha}{2(\alpha-1)}\log\left[\frac{\sum_{i\neq I}(\tilde{\phi}'_{ij}\tilde{h}_{ik})^2+(\tilde{\phi}'_{Ij}\tilde{h}_{Ik})^2}{\sum_{i\neq I}(\tilde{\phi}'_{ij}\tilde{h}_{ik})^2+(\phi'_{Ij}h_{Ik})^2}\right]\\
&\leq \frac{\alpha m}{2\sigma^2}\frac{n_\ell.\Delta_{\boldsymbol{f}}^2}{(\gamma_\ell^{min}\tau_h^{min})^2} + \frac{n_\ell.\alpha}{2(\alpha-1)}\log\left[\frac{\sum_{i\neq I}(\tilde{\phi}'_{ij}\tilde{h}_{ik})^2+(\gamma_\ell^{max}\tau_h^{max})^2}{\sum_{i\neq I}(\tilde{\phi}'_{ij}\tilde{h}_{ik})^2+(\gamma_\ell^{min}\tau_h^{min})^2}\right]\\
&\leq \frac{\alpha m}{2\sigma^2}\frac{n_\ell.\Delta_{\boldsymbol{f}}^2}{(\gamma_\ell^{min}\tau_h^{min})^2} + \frac{n_\ell.\alpha}{2(\alpha-1)}\log\left[\frac{(m-1).(\gamma_\ell^{min}\tau_h^{min})^2+(\gamma_\ell^{max}\tau_h^{max})^2}{(m-1).(\gamma_\ell^{min}\tau_h^{min})^2+(\gamma_\ell^{min}\tau_h^{min})^2}\right]\\
&\leq \frac{\alpha m}{2\sigma^2}\frac{n_\ell.\Delta_{\boldsymbol{f}}^2}{(\gamma_\ell^{min}\tau_h^{min})^2} + \frac{n_\ell.\alpha}{2(\alpha-1)}\log\left[\frac{(m-1).(\gamma_\ell^{min}\tau_h^{min})^2+(\gamma_\ell^{max}\tau_h^{max})^2}{m.(\gamma_\ell^{min}\tau_h^{min})^2}\right]\\
&\leq \frac{2.\alpha}{m.\sigma^2}\frac{(\gamma^{max}\tau^{max}\tau_B)^2}{(\gamma_\ell^{min}\tau_h^{min})^2} + \frac{n_\ell.\alpha}{2(\alpha-1)}\log\left[\frac{m-1}{m} + \frac{(\gamma_\ell^{max}\tau_h^{max})^2}{m.(\gamma_\ell^{min}\tau_h^{min})^2}\right].
\end{aligned}$$

6.6.2 Equal covariance matrices

First, we can notice that when the covariance matrices are equal, i.e. $\mathbf{\Sigma} = \mathbf{\Sigma}' = \frac{\sigma^2}{m^2}\mathbf{diag}(\boldsymbol{a}_k)^2$, the log-term in Equation 6.7 is equal to 0. Then, we have :

$$\begin{aligned}
\mathbb{D}_\alpha(P\|Q) &= \sum_{j=1}^{n_\ell} \frac{\alpha m^2}{2\sigma^2} \frac{(f_j(D) - f_j(D'))^2}{a_j^2} \\
&\leq \frac{n_\ell.\alpha.m}{2\sigma^2} \sum_{j=1}^{n_\ell} \frac{(f_j(D) - f_j(D'))^2}{(\gamma_\ell^{min}\tau_h^{min})^2} \\
&\leq \frac{n_\ell.\alpha.m}{2\sigma^2} \frac{\Delta_{\boldsymbol{f}}^2}{(\gamma_\ell^{min}\tau_h^{min})^2} \\
&\leq \frac{2\alpha}{m\sigma^2} \frac{(\tau_B\gamma_\ell^{max}\tau_h^{max})^2}{(\gamma_\ell^{min}\tau_h^{min})^2} \\
&\doteq \varepsilon_2.
\end{aligned}$$

6.6.3 Equal saturating covariance matrices

In this subsection, we will suppose that the covariance matrices are equal and saturating, i.e. $\mathbf{\Sigma} = \mathbf{\Sigma}' = \frac{\sigma^2}{n_\ell.m}(\gamma_\ell\tau_h)^2\mathbf{I}$ with $\gamma_\ell = \{\gamma_\ell^{min}, \gamma_\ell^{max}\}$ and $\tau_h = \{\tau_h^{min}, \tau_h^{max}\}$. Then we can start by noticing that $a_j^2 = a_j'^2 = \frac{m}{n_\ell}\tau_h^2\gamma_\ell^2$. In that case, the sensitivity of the function can be written as :

$$\begin{aligned}
\Delta_{\boldsymbol{f}}^\ell &= \sup_{D\sim D'} \|\boldsymbol{f}(D) - \boldsymbol{f}(D')\|_2 \leq \frac{2}{m} \left\|(\mathbf{B}^\ell \boldsymbol{e}_i) \odot \phi_\ell'(\mathbf{z}_i^\ell))h_{ik}^{\ell-1}\right\|_2 \\
&\leq \frac{2}{m}\tau_B\gamma_\ell\frac{\tau_h}{\sqrt{n_\ell}}.
\end{aligned}$$

This implies that :

$$\begin{aligned}
\mathbb{D}_\alpha(P\|Q) &= \sum_{j=1}^{n_\ell} \frac{\alpha m^2}{2\sigma^2} \frac{(f_j(D) - f_j(D'))^2}{a_j^2} \\
&\leq \frac{n_\ell.\alpha.m}{2\sigma^2} \sum_{j=1}^{n_\ell} \frac{(f_j(D) - f_j(D'))^2}{(\gamma_\ell\tau_h)^2} \\
&\leq \frac{n_\ell.\alpha.m}{2\sigma^2} \frac{\Delta_{\boldsymbol{f}}^2}{(\gamma_\ell\tau_h)^2} \\
&\leq \frac{2\alpha}{m\sigma^2} \frac{(\tau_B\gamma_\ell\tau_h)^2}{(\gamma_\ell\tau_h)^2} = \frac{2\alpha}{m\sigma^2}\tau_B^2 \\
&\doteq \varepsilon_3.
\end{aligned}$$

6.7 Appendix : additional numerical results on MNIST and CIFAR-10

MNIST – We provide below results on MNIST, obtained with the same code, procedure, and hyper-parameters as for the Fashion-MNIST experiments. These results are in line with Table 1

of our paper, with photonic results always close to ternarized ones. We notice that this "default" choice of hyper-parameters on MNIST results in ternarized DFA outperforming vanilla DFA. (We only lightly tune hyperparameters on BP, to demonstrate that our approach does not require any specific expensive fine-tuning search.)

σ / τ_B	**non-private**	**0.01**	**0.05**	**0.1**
			1	
BP	97,94	62,63	58,42	48,33
DFA	96,36	92,99	92,68	92,45
TDFA	97,09	93,67	93,57	93,28
PDFA	96,95	93,60	93,57	93,12

TABLE 6.2 – **Test accuracy on MNIST with our DP mechanism.** We find our approach to be robust to increasing DP noise σ. In particular, photonic DFA results (PDFA) are always within 1% of the corresponding DFA run.

CIFAR-10 – We chose to use a pre-trained network on ImageNet and extract its trained convolutional layers. Since these convolutions can be seen as feature extractors of the images and are not re-trained, they do not need to be taken into account into the Differential Privacy mechanism. We fine-tune only the fully-connected layers of the classifier using our Photonic DFA+DP mechanism.

We choose this experiment to demonstrate the scalability of our scheme. We do not seek to achieve state-of-the-art performance or to exhaustively explore the dynamics/impact of different differentially private configuration (as we did with MNIST), but simply to show our scheme can scale to such harder tasks.

We used a VGG16 network pre-trained on ImageNet. We leave the convolutions untouched, and fine-tune the classifier layers (25088 –> 4096 –> 4096 –> 10) with differentially private photonic training. We do not use any data augmentation, and simply resize the CIFAR-10 images to 224x224. We fine-tune for 15 epochs, using SGD with learning rate 5.10^{-3}, momentum 0.9, and batch size 256. Hyperparameters are kept identical across all methods and hardware. We obtain results both in a vanilla (no DP) setting as a comparison baseline, and in a differentially private setting yielding the following accuracies :

Vanilla (no differential privacy) : 83.17% (BP), 81.34% (DFA), 83.36% (TDFA).

DP ($\sigma = 0.05$, $\tau_f = 1$) : 60.45% (BP), 79.68% (DFA), 79.33% (TDFA), 78.64% (PDFA).

We note that this result shows good scalability, with performance in line with our MNIST and Fashion-MNIST results. Over all the experiments we have performed, the DFA algorithms seem much more resilient to adding noise and clipping (i.e. the DP algorithmical modification) than Backpropagation, which could open new research directions.

Chapter 7

Shedding a PAC-Bayesian Light on Adaptive Sliced-Wasserstein Distances

The Sliced-Wasserstein distance (SW) is a computationally efficient and theoretically grounded alternative to the Wasserstein distance. Yet, the literature on its statistical properties with respect to the distribution of slices, beyond the uniform measure, is scarce. To bring new contributions to this line of research, we leverage the PAC-Bayesian theory and the central observation that SW actually hinges on a slice-distribution-dependent Gibbs risk, the kind of quantity PAC-Bayesian bounds have been designed to characterize. We provide four types of results : i) PAC-Bayesian generalization bounds that hold on what we refer as *adaptive* Sliced-Wasserstein distances, i.e. distances defined with respect to any distribution of slices, ii) a procedure to learn the distribution of slices that yields a maximally discriminative SW, by optimizing our PAC-Bayesian bounds, iii) an insight on how the performance of the so-called *distributional* Sliced-Wasserstein distance may be explained through our theory, and iv) empirical illustrations of our findings.

7.1 Introduction

The Wasserstein distance is a metric between probability distributions and a key notion of the optimal transport framework [Villani, 2009 ; Peyré, 2019]. Over the past years, it has received a lot of attention from the machine learning community because of its theoretical grounding and the increasing number of problems relying on the computation of distances between measures [Solomon, 2014 ; Frogner, 2015 ; Montavon, 2016 ; Kolouri, 2017 ; Courty, 2016 ; Schmitz, 2018], such as the learning of deep generative models [Arjovsky, 2017 ; Bousquet, 2017 ; Tolstikhin, 2017]. As the measures μ and ν are usually unknown, the Wasserstein distance $\mathrm{W}(\mu, \nu)$ is estimated through an "empirical" version $\mathrm{W}(\mu_n, \nu_n)$, where $\mu_n \doteq \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and $\nu_n \doteq \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$ are i.i.d. samples (w.l.o.g. samples will be assumed to have the same size) from μ and ν, respectively. Due to its unfavorable $O(n^3 \log n)$ computational complexity, the plain Wasserstein distance

scales badly on large datasets ([Peyré, 2019]) and alternatives have been devised to overcome this limitation, such as the Sinkhorn algorithm [Cuturi, 2013 ; Cuturi, 2016], multi-scale [Oberman, 2015] or sparse approximations approaches [Schmitzer, 2016].

The Sliced-Wasserstein distance (SW) [Rabin, 2012] is another computationally efficient alternative, which takes advantage of the closed-form and fast computation of the one-dimensional Wasserstein distance. For d-dimensional (with $d > 1$) samples $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ and $\{\boldsymbol{y}_1, \dots, \boldsymbol{y}_n\}$, the computation of $\mathrm{SW}(\mu_n, \nu_n)$ is done by uniformly sampling L so-called *slices* $\{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_L\}$ and averaging the L one-dimensional Wasserstein $\mathrm{W}(\{\langle \boldsymbol{\theta}_j, \boldsymbol{x}_1\rangle, \dots, \langle \boldsymbol{\theta}_j, \boldsymbol{x}_n\rangle\}, \{\langle \boldsymbol{\theta}_j, \boldsymbol{y}_1\rangle, \dots, \langle \boldsymbol{\theta}_j, \boldsymbol{y}_n\rangle\})$ for $j = 1, \dots, L$. SW has been analyzed theoretically [Nadjahi, 2019 ; Nadjahi, 2020b], refined to gain additional efficiency [Nadjahi, 2021] or to handle nonlinear slices [Kolouri, 2019a ; Kolouri, 2020] and it has been successfully used in a variety of machine learning tasks [Bonneel, 2015 ; Kolouri, 2016 ; Carriere, 2017 ; Liutkus, 2019 ; Deshpande, 2018 ; Kolouri, 2018 ; Kolouri, 2019b ; Nadjahi, 2020a ; Bonet, 2021 ; Rakotomamonjy, 2021]. Another direction to improve SW consists in adapting the slice distribution in a data-dependent manner such as max-SW [Deshpande, 2019] which focuses on finding the unique slice $\boldsymbol{\theta}_\star$ (or equivalently, the Dirac measure $\delta_{\boldsymbol{\theta}_\star}$) that maximizes the Sliced-Wasserstein distance, or Distributional SW (DSW) [Nguyen, 2021], which seeks for a maximally discriminative distribution of slices on the unit sphere. A point untouched by those works, which fall into the class of what we refer as *adaptive* Sliced-Wasserstein distances and denote $\mathrm{SW}(\cdot, \cdot; \rho)$ by overloading notation and making the dependence on the slice distribution ρ clear, is the theoretical justification that the optimization procedures they each implement indeed guarantees generalization on unseen data. We here provide such an argument, in the form of a PAC-Bayesian bound, which covers all slice distributions, and connects the empirical Sliced-Wasserstein distance $\mathrm{SW}(\mu_n, \nu_n; \rho)$ with its population counterpart $\mathrm{SW}(\mu, \nu; \rho)$.

Three key reasons make the PAC-Bayesian theory, introduced in [McAllester, 1999] (see [Catoni, 2007 ; Alquier, 2021] for comprehensive reviews), particularly suited to characterize the generalization properties of adaptive SW. First, from a general perspective, the literature shows it allows the derivation of tight bounds converted into effective learning procedures [Ambroladze, 2007 ; Laviolette, 2006 ; Germain, 2009 ; Zantedeschi, 2021]. Second, PAC-Bayesian bounds deal with the generalization ability of learned distributions ; while those distributions usually are on spaces of predictors, the distributions ρ of interest in our case are the slice distributions. Lastly, a key quantity of PAC-Bayesian results, which is optimized when learning procedures are derived from bounds, is the empirical risk of the *stochastic Gibbs predictor*. The latter, when queried for a prediction for some input data, samples a predictor according to ρ and then outputs its prediction on the input data. As shown in the following, when the distribution ρ is the slice distribution, the empirical risk of the Gibbs predictor *exactly* boils down to the empirical adaptive $\mathrm{SW}(\mu_n, \nu_n; \rho)$ making the PAC-Bayesian theory the ideal framework to deal with SW. Slightly more formally, our results reads as : with probability $1 - \delta$, the following holds for all measures ρ on the d-dimensional unit sphere,

$$\mathrm{SW}(\mu, \nu; \rho) \geq \mathrm{SW}(\mu_n, \nu_n, \rho) - \varepsilon(n, \rho, \delta),$$

for some (small) ε that will be made explicit in our main Theorem 7.3.1.

Contributions and Outline. After recalling some essential notions in Section 7.2, we delve into our contributions : *i)* a PAC-Bayesian bound for *adaptive* Sliced-Wasserstein distances (Section 7.3), *ii)* a bound-optimizing procedure to train a maximally discriminative Sliced-Wasserstein distances (Section 7.4), *iii)* an uncovered connection between our work and the Distributional Sliced-Wasserstein distance of [Nguyen, 2021], bringing a piece to the missing theoretical ground supporting the latter approach (Section 7.4), and *iv)* empirical illustrations of the soundness of our theoretical results through experiments conducted both on toy and real-world datasets (Section 7.5).

7.2 Background

Notations. Let $\mathbb{N}^* \doteq \mathbb{N}\backslash\{0\}$ and $d \in \mathbb{N}^*$. $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^d$ and $\langle\cdot,\cdot\rangle$ the dot product. For some space $\mathsf{X} \subseteq \mathbb{R}^d$, $\mathcal{P}(\mathsf{X})$ is the set of probability distributions supported on X and $\mathcal{P}_q(\mathsf{X})$ the set of probability distributions supported on X with finite moment of order q. For $\mu \in \mathcal{P}(\mathsf{X})$, μ_n refers to the empirical measure supported on $n \in \mathbb{N}^*$ i.i.d. samples $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ from μ : $\mu_n \doteq n^{-1}\sum_{i=1}^n \delta_{\boldsymbol{x}_i}$, with δ_y the Dirac measure with mass on y —with a slight abuse of notation μ_n may also be used to refer to the corresponding n-sample. For $\mu \in \mathcal{P}(\mathbb{R})$, F_μ is the cumulative distribution function of μ and F_μ^{-1} its quantile function. $\mathcal{U}(\mathsf{X})$ denotes the uniform distribution on X.

7.2.1 The Sliced-Wasserstein distance

Wasserstein Distance. Let $p \in [1, +\infty)$, $\mu, \nu \in \mathcal{P}(\mathsf{X})$ and $\Pi(\mu,\nu) \subset \mathcal{P}(\mathsf{X}\times\mathsf{X})$ the set of distributions on $\mathsf{X}\times\mathsf{X}$ with marginals μ and ν. The Wasserstein distance of order p between μ and ν is

$$\mathrm{W}_p^p(\mu,\nu) \doteq \inf_{\pi\in\Pi(\mu,\nu)} \int_{\mathsf{X}\times\mathsf{X}} \|\boldsymbol{x}-\boldsymbol{y}\|^p \mathrm{d}\pi(\boldsymbol{x},\boldsymbol{y}). \tag{7.1}$$

W_p has been shown to possess appealing theoretical properties, but suffers from important computational limitations : solving Equation 7.1 may be costly, possibly suffering a worst-case $\mathcal{O}(n^3 \log n)$ complexity.

Fortunately, the Wasserstein distance is particularly easy to compute when the distributions to compare are univariate : for $\mu, \nu \in \mathcal{P}(\mathbb{R})$, there is a closed-form solution to Equation 7.1 given by,

$$\mathrm{W}_p^p(\mu,\nu) = \int_0^1 |F_\mu^{-1}(t) - F_\nu^{-1}(t)|^p \mathrm{d}t, \quad \mathrm{W}_p^p(\mu_n,\nu_n) = n^{-1}\sum_{i=1}^n |x_{(i)} - y_{(i)}|^p, \tag{7.2}$$

with $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$ and $y_{(1)} \leq y_{(2)} \leq \cdots \leq y_{(n)}$. Computing $\mathrm{W}_p(\mu_n,\nu_n)$ thus consists in sorting the support points of μ_n and ν_n, which induces $\mathcal{O}(n \log n)$ operations.

Sliced-Wasserstein Distance. SW defines a computationally efficient alternative to the Wasserstein distance, by leveraging the analytical solution of Equation 7.2 of $\mathrm{W}_p^p(\mu_n,\nu_n)$ between univariate distributions. Let $\mathbb{S}^{d-1} \doteq \{\boldsymbol{\theta} \in \mathbb{R}^d \ : \ \|\boldsymbol{\theta}\| = 1\}$ be the unit sphere in $\mathbb{R}^d$ and for $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$, denote by $\boldsymbol{\theta}^* : \mathbb{R}^d \to \mathbb{R}$ the linear map such that for $x \in \mathbb{R}^d$, $\boldsymbol{\theta}^*(\boldsymbol{x}) \doteq \langle\boldsymbol{\theta},\boldsymbol{x}\rangle$. Let $p \in [1,+\infty)$, $\mu,\nu \in \mathcal{P}(\mathsf{X})$ ($\mathsf{X} \subset \mathbb{R}^d$), and $\rho = \mathcal{U}(\mathbb{S}^{d-1})$. The Sliced-Wasserstein distance of order p

between μ, ν and based on ρ is defined as

$$\mathrm{SW}_p^p(\mu, \nu; \rho) \doteq \int_{\mathbb{S}^{d-1}} \mathrm{W}_p^p(\theta_\sharp^* \mu, \theta_\sharp^* \nu) \mathrm{d}\rho(\boldsymbol{\theta}), \tag{7.3}$$

where for any measurable function f and $\xi \in \mathcal{P}(\mathbb{R}^d)$, $f_\sharp \xi$ is the *push-forward measure* of ξ by f : for any measurable set A in $\mathbb{R}$, $f_\sharp \xi(\mathrm{A}) \doteq \xi(f^{-1}(\mathrm{A}))$ with $f^{-1}(\mathrm{A}) \doteq \{\boldsymbol{x} \in \mathbb{R}^d : f(\boldsymbol{x}) \in \mathrm{A}\}$. In particular, $\theta_\sharp^* \mu_n = n^{-1} \sum_{i=1}^n \delta_{\langle \boldsymbol{\theta}, \boldsymbol{x}_i \rangle}$ and $\theta_\sharp^* \nu_n = n^{-1} \sum_{i=1}^n \delta_{\langle \boldsymbol{\theta}, \boldsymbol{x}_i \rangle}$, so $\theta_\sharp^* \mu_n, \theta_\sharp^* \nu_n$ are the empirical measures of the data projected along $\boldsymbol{\theta}$, implying that $\mathrm{W}_p^p(\theta_\sharp^* \mu_n, \theta_\sharp^* \nu_n)$ has the analytical form of Equation 7.2. The Expectation of Equation 7.3 is commonly estimated with a Monte Carlo method and may provide significant speed-ups over the evaluation of $\mathrm{W}_p^p(\mu, \nu)$, thanks to the fast computation of $\mathrm{W}_p^p(\theta_\sharp^* \mu_n, \theta_\sharp^* \nu_n)$ [Bonneel, 2015].

Adaptive Sliced-Wasserstein Distance. Recent works have argued that $\rho = \mathcal{U}(\mathbb{S}^{d-1})$ is not necessarily the most relevant choice, and instead, proposed to adapt ρ to the data, providing $\mathrm{SW}_p^p(\cdot, \cdot; \rho)$ with an actual degree of freedom ρ and motivating the term of *adaptive* Sliced-Wasserstein distance. Specifically, [Deshpande, 2019] and [Nguyen, 2021] solve a tailored optimization problem in ρ targetting a high discriminative power of ρ, in the sense that ρ puts higher mass on the $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ that maximize the separation of $\theta_\sharp^* \mu$ and $\theta_\sharp^* \nu$: the *maximum Sliced-Wasserstein distance* (maxSW, [Deshpande, 2019]) and the *distributional Sliced-Wasserstein distance* (DSW, [Nguyen, 2021]) are defined as

$$\mathrm{maxSW}(\mu, \nu) \doteq \mathrm{SW}_p^p(\mu, \nu; \rho^\star_{\mathrm{maxSW}}(\mu, \nu)) \quad \text{and} \quad \mathrm{DSW}(\mu, \nu) \doteq \mathrm{SW}_p^p(\mu, \nu; \rho^\star_{\mathrm{DSW}}(\mu, \nu)) \tag{7.4}$$

where

$$\rho^\star_{\mathrm{maxSW}}(\mu, \nu) \doteq \underset{\delta_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \mathbb{S}^{d-1}}{\arg\sup} \mathrm{SW}_p^p(\mu, \nu; \delta_{\boldsymbol{\theta}}) \tag{7.5a}$$

$$\rho^\star_{\mathrm{DSW}}(\mu, \nu) \doteq \underset{\rho \in \mathcal{P}(\mathbb{S}^{d-1})}{\arg\sup} \mathrm{SW}_p^p(\mu, \nu; \rho) \quad s.t. \quad \mathbb{E}_{\boldsymbol{\theta}, \boldsymbol{\theta}' \sim \rho} |\boldsymbol{\theta}^\top \boldsymbol{\theta}'| \leq C, \tag{7.5b}$$

where, in Equation 7.5b, $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ are independent and $C > 0$ is a hyperparameter. We have decoupled the search for the maximizing arguments of Equation 7.5 and the maximum distances of Equation 7.4 for reasons we clarify below.

In practice, $\mathrm{maxSW}(\mu_n, \nu_n)$ and $\mathrm{DSW}(\mu_n, \nu_n)$ are computed instead of Equation 7.4 (that relies on the full measures μ and ν). While there exists statistical guarantees relating $\mathrm{maxSW}(\mu_n, \nu_n)$ and $\mathrm{maxSW}(\mu, \nu)$ [Lin, 2021] (likewise for DSW [Nguyen, 2021]), there is no theoretical argument, to the best of our knowledge, on the *test error* entailed by the learned distribution $\rho^\star_{\mathrm{maxSW}}(\mu_n, \nu_n)$ considered on its own, outside the optimization procedure of Equation 7.4 and Equation 7.5a of maxSW — hence the aforementioned decoupling. Given new samples $\{\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_n\}$ and $\{\boldsymbol{y}'_1, \ldots, \boldsymbol{y}'_n\}$ from μ and ν, with associated empirical distributions μ'_n and ν'_n, there is no guarantee for $\mathrm{SW}_p^p(\mu'_n, \nu'_n; \rho^\star_{\mathrm{maxSW}}(\mu_n, \nu_n))$ to be high, or in other words, there is no argument ensuring the discriminative power of $\rho^\star_{\mathrm{maxSW}}(\mu_n, \nu_n)$. One way to palliate this lack of theory and to go one step further than the maxSW and DSW cases, is to have at hand a general

result relating $\mathrm{SW}_p^p(\mu_n, \nu_n; \rho)$ and $\mathrm{SW}_p^p(\mu, \nu; \rho)$, for *any* $\rho \in \mathcal{P}(\mathbb{S}^{d-1})$. This is precisely what we bring in the following, in the form of a PAC-Bayesian generalization bound. More details about (Sliced) Optimal Transport are given in Section 2.7.

7.2.2 PAC-Bayes essentials

We here recall the core ingredients of the PAC-Bayesian theory [McAllester, 1999 ; Alquier, 2021 ; Catoni, 2007]. To this end, we take a little detour and consider a supervised learning setup : $S \doteq \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ is a training set of n i.i.d. pairs sampled from some unknown (and fixed) distribution $\mu \in \mathcal{P}(\mathcal{X} \times \mathcal{Y})$, where $\mathcal{X}$ is the space of features and $\mathcal{Y}$ is the target space, e.g. $\mathcal{Y} = \{-1, +1\}$. Given a *loss* function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, the empirical ℓ-risk $q_\ell(S, f)$ of a predictor $f \in \mathcal{Y}^{\mathcal{X}}$ $(= \{f, f : \mathcal{X} \to \mathcal{Y}\})$ is

$$q_\ell(S, f) \doteq n^{-1} \sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i), y_i). \tag{7.6}$$

PAC-Bayesian bounds provide statistical guarantees on the *true* ℓ-risk $R_\ell(\mu, \rho)$ of the Gibbs predictor $\rho \in \mathcal{P}(\mathcal{Y}^{\mathcal{X}})$ which, as said before, computes a prediction for input $\boldsymbol{x}$ in two steps : i) it samples f according to ρ and ii) it outputs $f(\boldsymbol{x})$. Defining the empirical ℓ-risk $r_\ell(S, \rho)$ of ρ and $R_\ell(\mu, \rho)$ as

$$r_\ell(S, \rho) \doteq \mathbb{E}_{f \sim \rho}\left[q_\ell(S, f)\right] \tag{7.7}$$

$$R_\ell(\mu, \rho) \doteq \mathbb{E}_{S \sim \mu^n} r_\ell(S, \rho) = \mathbb{E}_{f \sim \rho} \mathbb{E}_{(\boldsymbol{x}, y) \sim \mu} \ell(f(\boldsymbol{x}), y). \tag{7.8}$$

The following result from Catoni [Catoni, 2003 ; Alquier, 2021] epitomizes the nature of PAC-Bayesian bounds :

Theorem 7.2.1 (Catoni's bound [Catoni, 2003])**.** *Fix a* prior $\pi \in \mathcal{P}(\mathcal{Y}^{\mathcal{X}})$. $\forall \lambda > 0$, $\forall \delta \in (0, 1)$,

$$\mathbb{P}_{\mathcal{S}}\left[\forall \rho \in \mathcal{P}(\mathcal{Y}^{\mathcal{X}}), R_\ell(\mu, \rho) - r_\ell(S, \rho) \leq \frac{\lambda C^2}{8n} + \frac{KL(\rho||\pi) + \log \frac{1}{\delta}}{\lambda}\right] \geq 1 - \delta,$$

where $KL(\rho||\nu)$ *is the Kullback-Leibler divergence between* ρ *and* π, $KL(\rho||\nu) \doteq \int \log\left(\frac{\mathrm{d}\rho}{\mathrm{d}\pi}(\boldsymbol{\theta})\right) \rho(\mathrm{d}\boldsymbol{\theta})$ *(defined if* ρ *has a density* $\mathrm{d}\rho/\mathrm{d}\pi$ *with respect to* π*).*

In the following, we will develop such type of bound for *adaptive* Sliced-Wasserstein distances. More details about the PAC-Bayesian framework are given in Section 2.6.

7.3 PAC-Bayes Generalization Bounds for Adaptive Sliced-Wasserstein

We introduce new results on the generalization properties of adaptive Sliced-Wasserstein distances (all proofs are deferred to the Appendix). Specifically, we leverage the PAC-Bayes framework to bound the error induced by the approximation of $\mathrm{SW}_p^p(\mu, \nu; \rho)$ by $\mathrm{SW}_p^p(\mu_n, \nu_n; \rho)$ for any

$\rho \in \mathcal{P}(\mathbb{S}^{d-1})$. These results, which apply to specific settings directed by conditions on the supports and the moments of μ and ν, are specializations of our generic following theorem :

Theorem 7.3.1. *Let $p \in [1, +\infty)$ and $\mu, \nu \in \mathcal{P}_p(\mathbb{R}^d)$. Assume there exists a constant $\varphi_{\mu,\nu,p}$ possibly depending on μ, ν and p, such that for $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ and $\lambda > 0$,*

$$\mathbb{E}\left[\exp\left(\lambda\Big\{\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n) - \mathbb{E}[\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n)]\Big\}\right)\right] \leq \exp(\lambda^2 \varphi_{\mu,\nu,p} n^{-1}), \tag{7.9}$$

where $\mathbb{E}$ is taken with respect to the support points of μ_n, ν_n. Additionally, assume there exists a function $\psi_{\mu,\nu,p}$ of n, possibly depending on μ, ν and p, such that

$$\mathbb{E}|\mathrm{SW}_p^p(\mu_n, \nu_n; \rho) - \mathrm{SW}_p^p(\mu, \nu; \rho)| \leq \psi_{\mu,\nu,p}(n). \tag{7.10}$$

Let $\rho_0 \in \mathcal{P}(\mathbb{S}^{d-1})$ and $\delta > 0$. Then, with probability at least $1-\delta$, the following holds : $\forall \rho \in \mathcal{P}(\mathbb{S}^{d-1})$

$$\mathrm{SW}_p^p(\mu_n, \nu_n; \rho) \leq \mathrm{SW}_p^p(\mu, \nu; \rho) + \lambda\varphi_{\mu,\nu,p}/n + \{KL(\rho||\rho_0) + \log(1/\delta)\}/\lambda + \psi_{\mu,\nu,p}(n). \tag{7.11}$$

Remark 7.3.2 (Elements of proof)**.** *This theorem is partly an instantiation of Theorem 7.2.1 and the accompanying quantities of Equation 7.6, Equation 7.7 and Equation 7.8, where predictors are here replaced by slices, and ρ and ρ_0 are distributions on slices. The key observation stressed out from the start can now be made formal : $q_\ell(S, f)$ of Equation 7.6 is here replaced by $\mathrm{SW}_p^p(\mu_n, \nu_n; \delta_{\boldsymbol{\theta}})$ for some $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$, and $r_\ell(S, \rho)$ of Equation 7.7 by $\mathbb{E}_{\boldsymbol{\theta}\sim\rho}\mathrm{SW}_p^p(\mu_n, \nu_n; \delta_{\boldsymbol{\theta}}) = \mathrm{SW}_p^p(\mu_n, \nu_n; \rho)$. Also, a key tool to obtain the desired results will be the use of (a refined version of) McDiarmid [McDiarmid, 1989] inequality that can state concentration results on $\mathrm{SW}_p^p(\mu_n, \nu_n; \delta_{\boldsymbol{\theta}})$ (with respect to $\{\boldsymbol{x}_i\}_{i=1}^n$ and $\{\boldsymbol{y}_i\}_{i=1}^n$) — invoking bounded-difference properties.*

Theorem 7.3.1 guarantees that the gap $\mathrm{SW}_p^p(\mu_n, \nu_n; \rho) - \mathrm{SW}_p^p(\mu, \nu; \rho)$ is controlled by the moments of $\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n) - \mathbb{E}[\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n)]$ for any $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$, the convergence rate of $\mathbb{E}\mathrm{SW}_p^p(\mu_n, \nu_n; \rho)$ towards $\mathrm{SW}_p^p(\mu, \nu; \rho)$, and the dissimilarity between ρ and ρ_0. The bound Equation 7.11 has a $O(n^{-\frac{1}{2}})$ convergence rate when $\lambda = n^{\frac{1}{2}}$ which is a standard rate in learning theory. However, as in [Haddouche, 2021], we could parameterize $\lambda = n^\alpha$ and perform an alternate optimization of the bound over ρ and α.

Remark 7.3.3. *Since this bound is for all $\rho \in \mathcal{P}(\mathbb{S}^{d-1})$, it is therefore valid for the distribution $\rho^\star_{maxSW}$ of Equation 7.5a and $\rho^\star_{DSW}$ of Equation 7.5b computed by maxSW and DSW.*

Our bound Equation 7.11 can be refined under various assumptions on the supports and moments of μ and ν. We explore these assumptions and observe the entailed bounds : first we suppose the supports are bounded (Section 7.3.1) ; then, we only assume μ, ν are sub-Gaussian or satisfy a Bernstein-type moment condition (Section 7.3.2). We show in the next sections that $\varphi_{\mu,\nu,p}$ and $\psi_{\mu,\nu,p}$ can be computed explicitly in each of the three settings so that Theorem 7.3.1 can be specialized.

7.3.1 Distributions supported on bounded domains

In our first setting, we study distributions with bounded supports, as formally stated in A1.

A 1. $\mu, \nu \in \mathcal{P}(\mathsf{X})$ *where* $\mathsf{X} \subset \mathbb{R}^d$ *has a finite diameter* $\Delta = \sup_{(\boldsymbol{x},\boldsymbol{x}')\in\mathsf{X}^2} \|\boldsymbol{x} - \boldsymbol{x}'\| < +\infty$.

Under A1, we can apply similar arguments as in the proof of McDiarmid's inequality [McDiarmid, 1989] to derive $\varphi_{\mu,\nu,p}$. This yields Proposition 7.3.4, which can be seen as a particular instance of [Weed, 2019, Proposition 20].

Proposition 7.3.4. *Let* $p \in [1, +\infty)$ *and assume A1. Then,* $\varphi_{\mu,\nu,p} = \Delta^{2p}/4$.

We emphasize that Proposition 7.3.4 follows from the proof of [Weed, 2019, Proposition 20], which leverages McDiarmid's inequality to establish a concentration bound for $\mathrm{W}_p^p(\mu, \mu_n)$ around its expectation on any finite-dimensional compact spaces.
Next, we adapt the proof of [Manole, 2020, Lemma B.3] to compute the explicit form of $\psi_{\mu,\nu,p}$.

Proposition 7.3.5. *Let* $p \in [1, +\infty)$ *and assume A1. There exists* $C(p) > 0$ *depending on* p *such that*

$$\psi_{\mu,\nu,p}(n) = C(p)\Delta^{p-1}\{SJ(\mu;\rho) + SJ(\nu;\rho)\}n^{-1/2},$$

where for $\xi \in \{\mu, \nu\}$, $SJ(\xi;\rho) = \int_{\mathbb{S}^{d-1}} \int_{-\infty}^{+\infty} \{F_{\theta^*_\sharp\xi}(t)(1 - F_{\theta^*_\sharp\xi}(t))\}^{1/2} \mathrm{d}t\mathrm{d}\rho(\boldsymbol{\theta})$.

Proposition 7.3.5 shows that the expected approximation error $\mathbb{E}|\mathrm{SW}_p^p(\mu_n, \nu_n; \rho) - \mathrm{SW}_p^p(\mu, \nu; \rho)|$ decays at the rate $n^{-1/2}$, provided that $\{SJ(\mu;\rho) + SJ(\nu;\rho)\} < +\infty$. By [Bobkov, 2019, Section 3.1], this functional is indeed finite assuming A1.
Combining Propositions 7.3.4 and 7.3.5 makes it possible to specialize Theorem 7.3.1 to distributions supported on bounded domains. The resulting bound is given in Appendix 7.7.4.

7.3.2 Distributions supported on unbounded domains

We extend our analysis to distributions with unbounded supports. To handle this case, we will assume additional constraints on the moments on μ, ν, which will allow the application of generalized McDiarmid's inqualities to derive $\varphi_{\mu,\nu,p}$. We consider the following two settings : distributions are *sub-Gaussian* (A2) or satisfy a *Bernstein-type moment condition* (A3).
Before deriving $\varphi_{\mu,\nu,p}$ for Sub-Gaussian distributions, let us formally recall their characterization.

Definition 7.3.6 (Sub-Gaussian distribution)**.** *Let* $\mu \in \mathcal{P}(\mathbb{R}^d)$ *and* $\sigma > 0$. μ *is a* sub-Gaussian distribution *with* variance proxy σ^2 *if for any* $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$, *for* $\lambda \in \mathbb{R}$, $\int_{\mathbb{R}} \exp(\lambda t)\mathrm{d}(\theta^*_\sharp\mu)(t) \leq \exp(\lambda^2\sigma^2/2)$.

A 2. $\mu, \nu \in \mathcal{P}(\mathbb{R}^d)$ *are sub-Gaussian with respective variance proxies* σ^2, τ^2 *;* $\hat{\sigma}^2$ *and* $\hat{\tau}^2$ *are the variance proxies of* $\{\mu_n\}_{n\in\mathbb{N}^*}, \{\nu_n\}_{n\in\mathbb{N}^*}$ *(which exist almost surely [Mena, 2019, Lemma A.2]).*

Under A2, we apply the generalized McDiarmid's inequality for unbounded spaces with finite *sub-Gaussian diameter* ([Kontorovich, 2014], recalled in the Appendix) to derive $\varphi_{\mu,\nu,1}$.

Proposition 7.3.7. *Under A2,* $\varphi_{\mu,\nu,1} = \hat{\sigma}^2 + \hat{\tau}^2$.

	Bounded supports (A1)	Unbounded supports	
		Sub-Gaussianity (A2)	Bernstein moments (A3)
$\varphi_{\mu,\nu,p}$	Proposition 7.3.4	Proposition 7.3.7	Proposition 7.3.9
$\psi_{\mu,\nu,p}$	Proposition 7.3.5	[Manole, 2020]	

TABLE 7.1 – Overview of the explicit forms of $\varphi_{\mu,\nu,p}$ and $\psi_{\mu,\nu,p}$ under different assumptions.

We move on to our second setting for distributions with unbounded supports, which relaxes A2 by assuming a Bernstein-type moment condition instead, as described hereafter.

Definition 7.3.8 (Bernstein condition)**.** *Let $\sigma^2, b > 0$ and $\mu \in \mathcal{P}(\mathbb{R}^d)$; μ is said to satisfy the* (σ^2, b)-Bernstein condition *if for any $k \in \mathbb{N}$, $k \geq 2$, $\int_{\mathbb{R}^d} \|\boldsymbol{x}\|^k \mathrm{d}\mu(\boldsymbol{x}) \leq \sigma^2 k! b^{k-2}/2$.*

A 3. *Let $\mu, \nu \in \mathcal{P}(\mathbb{R}^d)$ be two distributions satisfying the Bernstein condition with parameters $(\sigma^2, b), (\tau^2, c)$ respectively.*

Note that Definition 7.3.6 is strictly stronger than Definition 7.3.8 : if $\mu \in \mathcal{P}(\mathbb{R}^d)$ verifies the (σ^2, b)-Bernstein condition, then μ belongs to the class of heavy-tailed distributions called *sub-exponential distributions* [Embrechts, 2013], which contains sub-Gaussian distributions. Hence, assuming A3 leads to a larger class of distributions than A2. Under A3, one can explicitly compute $\varphi_{\mu,\nu,1}$ by using a Bernstein-type McDiarmid's inequality : the proof of [Lei, 2020, Corollary 5.2] yields the next proposition.

Proposition 7.3.9. *Assume A3 and let $\sigma_\star^2 = \max(\sigma^2, \tau^2)$, $b_\star = \max(b, c)$. Then, for $\lambda > 0$ s.t. $\lambda < (2b_\star)^{-1} n$, $\varphi_{\mu,\nu,1} = 2\sigma_\star^2 n^{-1}(1 - 2b_\star \lambda n^{-1})^{-1}$.*

The last ingredient to specify Theorem 7.3.1 under A2 and A3 is to derive $\psi_{\mu,\nu,p}$. Indeed, since the supports of μ, ν are unbounded, the necessary condition for the finiteness of $\{SJ(\mu;\rho) + SJ(\nu;\rho)\}$ is not met, thus deteriorating the rate of convergence in Proposition 7.3.5. To overcome this issue, we will use the rate recently established in [Manole, 2020], which holds true on A2 and A3 and shows that $\psi_{\mu,\nu,p}$ scales as $n^{-1/2}\log(n)$. Our final bound is obtained by plugging in Theorem 7.3.1 the explicit formula of $\psi_{\mu,\nu,1}$ and Propositions 7.3.7 and 7.3.9 : we present this result and its detailed proof in Appendix 7.7.7.

We conclude this section by summarizing in Table 7.1 the setups for which we can explicitly compute $\varphi_{\mu,\nu,p}$ and $\psi_{\mu,\nu,p}$, thus yielding refined versions of our generic bound in Equation 7.11. Note that on unbounded supports, we derived $\varphi_{\mu,\nu,p}$ for $p = 1$ only : the generalized McDiarmid's inequalities leading to Propositions 7.3.7 and 7.3.9 can be applied if W_p^p is Lipschitz [Kontorovich, 2014 ; Lei, 2020], which is easily verified for $p = 1$ but not for $p > 1$. Hence, the derivation of $\varphi_{\mu,\nu,p}$ for $p > 1$ and μ, ν supported on unbounded domains requires different proof techniques. We leave this problem for future work.

7.4 Applications

We develop a methodology to find the distribution in $\mathbb{S}^{d-1}$ which optimizes our bounds in Section 7.3 so that besides generalizing well to unseen data, SW based on that distribution is

highly discriminative. We then connect our contributions to prior related work : we show that our theoretical findings shed light on the Distributional Sliced-Wasserstein distance. In what follows, $\sigma = \mathcal{U}(\mathbb{S}^{d-1})$.

7.4.1 Optimization of our bound

Optimizing Equation 7.11 boils down to maximizing the terms dependent on ρ (when λ is set to $n^{-\frac{1}{2}}$), leading to the following problem :

$$\rho^\star(\mu_n, \nu_n) = \underset{\rho \in \mathcal{P}(\mathbb{S}^{d-1})}{\arg\sup}\ \mathrm{SW}_p^p(\mu_n, \nu_n; \rho) - \mathrm{KL}(\,\rho||\sigma)/n^{1/2}. \tag{7.12}$$

Theorem 7.3.1 guarantees with high probability that solving Equation 7.12 results in the maximization of $\mathrm{SW}_p^p(\mu, \nu; \rho)$ over $\rho \in \mathcal{P}(\mathbb{S}^{d-1})$. In that sense, and according to our theory, $\rho^\star(\mu_n, \nu_n)$ has an important discriminative power : our generalization bound implies that $\mathrm{SW}_p^p(\mu, \nu; \rho^\star(\mu_n, \nu_n))$ has a large value, even though $\rho^\star$ is learned from limited samples from μ, ν.
We propose to compute $\rho^\star$ as the solution of a parametric optimization problem, using gradient-based optimization. The candidate solutions in Equation 7.12 are either parameterized as *von Mises-Fisher distributions*, a class of distributions on $\mathbb{S}^{d-1}$ (Definition 7.4.1) or as $f_\sharp\sigma$, with f a neural network. In the latter case, the KL divergence has no analytical form, and we approximate it using [Ghimire, 2021]. Our final procedure is depicted in Algorithm 4.

Algorithm 4 PAC-SW : Adaptive SW via PAC-Bayes bound optimization

Input : empirical distributions (μ_n, ν_n), initialized distribution $\rho^{(0)}$, maximum number of iterations T, learning rate η
for $t \leftarrow 1$ to T **do**
  $\mathcal{L}(\mu_n, \nu_n; \rho^{(t-1)}) = \mathrm{SW}_p^p(\mu_n, \nu_n; \rho^{(t-1)}) - \mathrm{KL}(\rho^{(t-1)}||\sigma)/n^{1/2}$
  $\rho^{(t)} = \rho^{(t-1)} + \eta \nabla_\rho \mathcal{L}(\mu_n, \nu_n; \rho^{(t-1)})$
end for
return $\rho^{(T)}$

7.4.2 Theoretical insights on the Distributional Sliced-Wasserstein distance

We now use our theoretical findings to shed light on the practical performance of DSW, by connecting our procedure with the optimization problem underlying DSW. To achieve this, we study a specific instance of DSW, for which the optimization is performed over a family of parametric distributions on $\mathbb{S}^{d-1}$ called von Mises-Fisher distributions (vMF, Definition 7.4.1). We demonstrate that in that case, the penalization term is equal, up to an affine transformation, to the KL term in Equation 7.12. We rigorously prove this result, then give some intuition on why the connection between DSW and our adaptive SW carries over when the search space is parametrized by a neural network, as in [Nguyen, 2021].

Definition 7.4.1. *The* von Mises-Fisher distribution *with* mean direction $\mathbf{m} \in \mathbb{S}^{d-1}$ *and* concentration parameter $\kappa \in \mathbb{R}_+^*$, *denoted by* $vMF(\mathbf{m}, \kappa)$, *is a distribution on* $\mathbb{S}^{d-1}$ *whose density is defined for* $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ *by* $vMF(\boldsymbol{\theta}; \mathbf{m}, \kappa) = C_{d/2}(\kappa)\exp(\kappa \mathbf{m}^\top \boldsymbol{\theta})$, *where* $C_{d/2}(\kappa) =$

$\kappa^{d/2-1}/\{(2\pi)^{d/2} I_{d/2-1}(\kappa)\}$ *and* $I_{d/2-1}$ *is the* modified Bessel function of the first kind at order $d/2-1$.

Intuitively, the higher κ, the more concentrated $\text{vMF}(\mathbf{m}, \kappa)$ is around $\mathbf{m}$. Von Mises-Fisher distributions have been successfully deployed in several statistical and machine learning problems to effectively model spherical data [Hasnat, 2017; Kumar, 2018; Scott, 2021]. We propose to use vMF to parameterize the search space of Equation 7.5b and Equation 7.12, *i.e.* the argmax is over $\rho \in \{\text{vMF}(\mathbf{m}, \kappa), \mathbf{m} \in \mathbb{S}^{d-1}, \kappa \in \mathbb{R}_+^*\}$. The regularization in Equation 7.5b implies $\mathbb{E}_{\boldsymbol{\theta}, \boldsymbol{\theta}' \sim \text{vMF}(\mathbf{m}, \kappa)}[\boldsymbol{\theta}^\top \boldsymbol{\theta}'] \leq C$. Since $\mathbb{E}_{\boldsymbol{\theta} \sim \text{vMF}(\mathbf{m}, \kappa)}[\boldsymbol{\theta}] = \{I_{d/2}(\kappa)/I_{d/2-1}(\kappa)\}\mathbf{m}$, one can show that $\mathbb{E}_{\boldsymbol{\theta}, \boldsymbol{\theta}' \sim \text{vMF}(\mathbf{m}, \kappa)}[\boldsymbol{\theta}^\top \boldsymbol{\theta}'] = \{I_{d/2}(\kappa)/I_{d/2-1}(\kappa)\}^2$. On the other hand, by [Davidson, 2018],

$$\text{KL}(\text{vMF}(\mathbf{m}, \kappa)||\sigma) = \kappa I_{d/2}(\kappa)/I_{d/2-1}(\kappa) + \log C_{d/2}(\kappa) + \log\left(2\pi^{d/2}/\Gamma(d/2)\right). \tag{7.13}$$

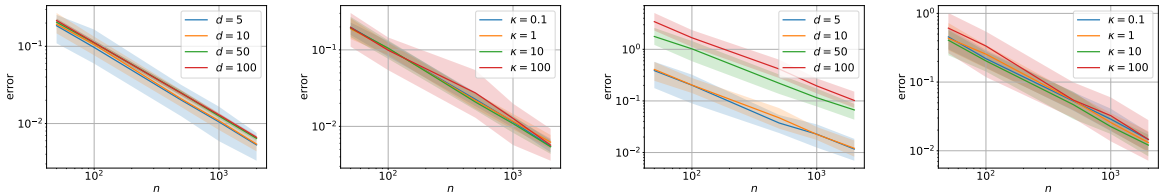
By Equation 7.13, the regularization term in Equation 7.12 is a function of the penalization in DSW. In that sense, computing DSW boils down to optimizing our bound, up to some factors depending on κ, d and n.
In the more general setting studied in [Nguyen, 2021], the derivation of an analytical expression relating DSW and our strategy is difficult, since the candidate solutions are defined as $f_\sharp \sigma$ with f being a deep neural network. Instead, we conjecture that computing DSW in that setup still amounts to optimizing an incomplete version of our bound, based on the following argument : the regularization in Equation 7.5b controls the concentration of the distribution over the slices [Nguyen, 2021, Section 3.1] ; more precisely, the higher $\mathbb{E}|\boldsymbol{\theta}^\top \boldsymbol{\theta}'|$ with $\boldsymbol{\theta}, \boldsymbol{\theta}'$ i.i.d. from $f_\sharp \sigma$, the more concentrated $f_\sharp \sigma$ is around specific directions, so the higher $\text{KL}(f_\sharp \sigma||\sigma)$. We will conduct an empirical analysis in Section 7.5 to verify our intuition.

7.5 Numerical experiments

We now conduct an empirical analysis to confirm our theoretical contributions and illustrate their consequences in practice. We first validate Theorem 7.3.1 and its refinements, by studying the influence of the terms appearing in the bound. Then, we compare DSW and Algorithm 4, to confirm the connection drawn in Section 7.5.2. Finally, we demonstrate the discriminative power of the distributions of slices learned by Algorithm 4 on both synthetic and real data. In particular, we show our strategy can be leveraged to reduce the training time when training generative models. All experiments were run on a GPU NVIDIA V100 32GB and are reproducible using the code in https://github.com/rubenohana/PAC-Bayesian_Sliced-Wasserstein.

7.5.1 Empirical confirmation of our bounds

Our first set of experiments aims at confirming the theoretical bounds derived in Section 7.3. We sample two sets of n i.i.d. samples from the same distribution $\mu \in \mathcal{P}(\mathbb{R}^d)$. To illustrate our bound on both bounded and unbounded supports, we choose μ as a uniform or Gaussian distribution. We approximate $\text{SW}_p^p(\mu_n, \nu_n; \text{vMF}(\mathbf{m}, \kappa))$ with $\mathbf{m}$ picked uniformly on $\mathbb{S}^{d-1}$ and $\kappa > 0$, by computing its Monte Carlo estimate over 1000 projection directions. Figure 7.1 plots



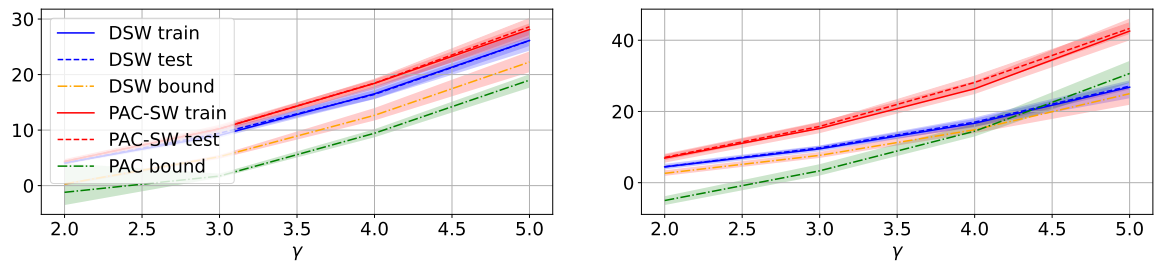
(a) Uniform distribution (b) Gaussian distribution

FIGURE 7.1 – $\mathrm{SW}_p^p(\mu_n, \nu_n; \mathrm{vMF}(\mathbf{m}, \kappa))$ vs. n for $\mu = \nu = \mathcal{U}([0,5]^d)$ (7.1a) or $\mathcal{N}(\mathbf{0}, \Sigma_d)$ (7.1b). Results are averaged over 30 runs and shown on log-log scale, with their 10th-90th percentiles.

the approximation error (which here, reduces to $\mathrm{SW}_p^p(\mu_n, \nu_n; \mathrm{vMF}(\mathbf{m}, \kappa))$ since the two datasets come from the same distribution) against n for different values of d and κ. We observe that the error decays to 0 as n increases, and the convergence rate is slower as d and κ increase. This is consistent with our theoretical analysis : the higher d, the larger the diameter (respectively, the sub-Gaussian diameter) when μ is uniform (resp., Gaussian), so the larger $\varphi_{\mu,\nu,p}$ (Propositions 7.3.4 and 7.3.7) ; the higher κ, the larger $\mathrm{KL}(\mathrm{vMF}(\mathbf{m}, \kappa)||\sigma)$.

7.5.2 Comparison with the Distributional Sliced-Wasserstein distance

Next, we illustrate the relation between DSW and PACSW, as established in Section 7.4.2, where ρ is parameterized as a vMF distribution. Here, we compare $\mu = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_d)$ and $\nu = \mathcal{N}(\gamma \mathbf{1}, \mathbf{\Sigma}_d)$, with $\gamma > 0$, $\mathbf{\Sigma}_d \in \mathbb{R}^{d \times d}$ symmetric positive semi-definite set at random, and $\mathbf{0}$ (resp., $\mathbf{1}$) the vector whose components are all equal to 0 (resp., 1). The higher γ, the more dissimilar μ and ν. We sample $n = 500$ samples from μ and ν and compute $\rho^\star_{\mathrm{DSW}}(\mu_n, \nu_n)$ and $\rho^\star(\mu_n, \nu_n)$. The optimization is performed on the space of vMF distributions, using Adam [Kingma, 2015] with its default parameters. To analyze the generalization properties of adaptive SW, we sample $m = 2000$ test points from μ, ν and compute $\mathrm{SW}_p^p(\mu_m, \nu_m; \rho^\star_{\mathrm{DSW}}(\mu_n, \nu_n))$ and $\mathrm{SW}_p^p(\mu_m, \nu_m; \rho^\star(\mu_n, \nu_n))$. Results for different values of d and γ are reported in Figure 7.2, and confirm the generalization ability of both $\rho^\star(\mu_n, \nu_n)$ and $\rho^\star_{\mathrm{DSW}}(\mu_n, \nu_n)$. Besides, we observe that while our strategy returns the most discriminative distribution, since the values of SW are the highest, DSW follows closely. This confirms the connection with DSW.



(a) $d = 5$ (b) $d = 20$

FIGURE 7.2 – Comparison of DSW and PAC-SW for $\mu = \mathcal{N}(\mathbf{0}, \Sigma_d)$ and $\nu = \mathcal{N}(\gamma \mathbf{1}, \Sigma_d)$. Results are averaged over 10 runs and reported with their 10th-90th percentiles.

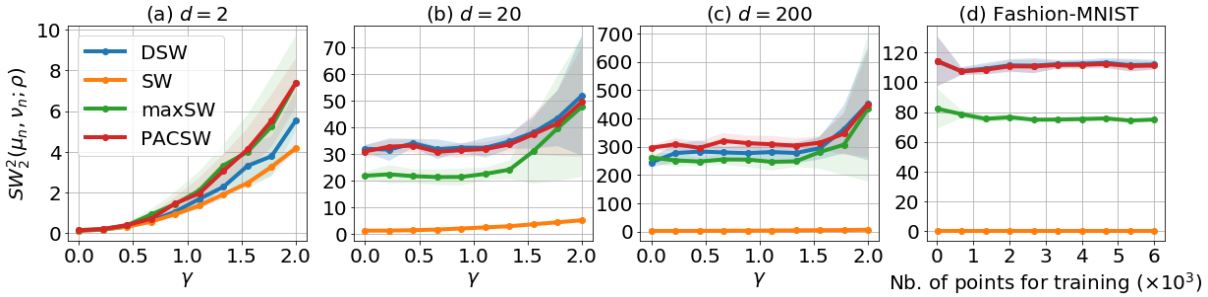


FIGURE 7.3 – $\mathrm{SW}_p^p(\mu_n, \nu_n; \rho)$ with (a-c) $\mu = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_d)$ and $\nu = (\gamma \mathbf{1}, \mathbf{\Sigma}_2)$ and $n = 1000$; as a function of γ, (d) classes 4 and 5 of Fashion-MNIST; as a function of the number of training points. ρ is learned on the train set, and we report values on the test set.

7.5.3 Illustration of the generalization ability of Adaptive Sliced-Wasserstein distances

Our goal is to verify the generalization ability of adaptive SW distances for evaluating the distance between two data distributions with ρ parametrized as a neural network, as in Section 7.4.1. In each experiment, we learn the optimal distribution ρ for each algorithm on a *training set*, and we observe how this learned distribution performs on a *test set* of the same size. We consider 200 slices, the learning rate η is taken as the best (i.e. yielding the higher distance) out of $[10^{-3}, 10^{-2}, 10^{-1}, 1]$. Each run is averaged 10 times with standard deviations in shaded areas. On Figure 7.3(a-c), we measure the distance between two Gaussians, as in Section 7.5.2. We can observe that PACSW is always amongst the most discriminative distances. On Figure 7.3(d), we measure the distance between 2 classes of the Fashion-MNIST dataset [Xiao, 2017] – classes 4 (*coats*) and 5 (*sandals*), $d = 784$, images rescaled between 0 and 1 – and vary the number of training points. We observe on this plot that PACSW and DSW ($\lambda_C = 10$) return higher values than maxSW and SW based on uniform sampling, which shows that they are able to better discriminate data and generalize well.

7.5.4 Illustrating generalization properties on generative modelling

In this experiment, we illustrate the generalization properties of DSW, which is encompassed in our theory and has been shown to be strongly linked to our distance in Section 7.4.2. We choose to focus on DSW because it is more tractable than our distance in a large-scale setting, as it uses a rough but computationally cheap approximation of the KL divergence.

The generalization properties are evaluated on a MNIST generative modelling context and we compare the performance of a model trained using DSW as a loss, in the flavor of [Deshpande, 2018]. Usually, the distribution of slices is learned at every minibatch of data. However, we make the hypothesis that if the learned distribution generalizes well to unseen datasets, then gradients obtained from the distance between minibatches would still provide sufficient information to learn the generative model. As a consequence, we evaluate the robustness and generalization ability of the learned distribution using our PAC approach by learning it only every 10 or 50 minibatches (denoted by -10 or -50 resp.).

For training the model, we followed the same approach (architecture and optimizer) as the one described in [Nguyen, 2021]. For each minibatch of size 512, the distribution ρ is learned by optimizing 100 projections over 100 iterations and the generative model is trained over 400 epochs. For a sake of comparison, we report also results of a generative model trained with maxSW.

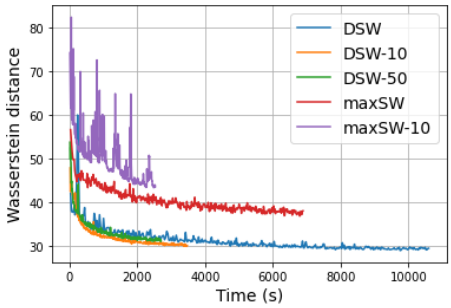


FIGURE 7.4 – Evolution of the Wasserstein distance between a set of generated MNIST digits and the true MNIST test set with respect to training time.

Figure 7.4 shows the evolution of the Wasserstein distance (WD) between generated data and the test set with respect to training time (measured after each epoch), for each distance and different update rate of the distribution ρ. We can observe that classical DSW yields a WD of 29 after $\sim 10^4$s. When learning ρ every 10 minibatch (DSW-10), we achieve similar a WD value with half the running time. When further reducing the frequency update of ρ (DSW-50), we converge faster but with a loss in quality of generation (WD ~ 32). Our result on maxSW confirms our theoretical analysis because in many learning problems, regularization is key to help generalize on unseen data. Indeed, while using maxSW as a loss yields a reasonable performance, optimizing the slice every 10 minibatches leads to a very unstable learning and worst performances. Examples of generated digits are in Appendix 7.8.2.

7.6 Conclusion

This work addresses the question of the generalization properties of adaptive Sliced-Wasserstein distances, i.e. Sliced-Wasserstein distances whose slice distribution may be different from the classical uniform distribution. We introduce a generic PAC-Bayesian result that characterizes these generalization abilities and refine it according to additional constraints on the moment and support properties of the distributions μ and ν to be compared. We build a learning algorithm from this theoretical findings, which essentially consists in optimizing the generic bound. When run on toy models and generative modelling, our procedure shows to be effective, and to compare favourably with the related maxSW and the Distributional Sliced-Wasserstein distances. As it happens, we show that our theory provides clues on why the latter distance is effective in practice. Extensions of this work would be to come up with other training algorithms of slice distributions and to see how our theory could be extended to other sliced distances [Paty, 2019 ; Nadjahi, 2020b].

7.7 Appendix : postponed proofs for Section 7.3

7.7.1 Proof of Theorem 7.3.1

Theorem 7.3.1 is obtained by adapting standard results in the literature on PAC-Bayes bounds, and can actually be seen as a particular case of Catoni's bound [Catoni, 2003], which, to the best of our knowledge, have never been studied in prior work. We provide the detailed proof for completeness.
First, we recall Donsker and Varadhan's variational formula, which plays a central role in the PAC-Bayesian framework.

Lemma 7.7.1 (Donsker and Varadhan's variational formula [Donsker, 1975])**.** *Let $\boldsymbol{\theta}$ be a set equipped with a σ-algebra and $\pi \in \mathcal{P}(\Theta)$. For any measurable, bounded function $h : \boldsymbol{\theta} \to \mathbb{R}$,*

$$\log \mathbb{E}_{\boldsymbol{\theta}\sim\pi}\big[\exp(h(\boldsymbol{\theta}))\big] = \sup_{\rho\in\mathcal{P}(\Theta)} \big[\mathbb{E}_{\boldsymbol{\theta}\sim\rho}[h(\boldsymbol{\theta})] - KL(\rho||\pi)\big]. \tag{7.14}$$

Proof of Theorem 7.3.1. Let $p \in [1, +\infty)$ and $\mu, \nu \in \mathcal{P}_p(\mathbb{R}^d)$. Assume there exists $\varphi_{\mu,\nu,p}$ such that for any $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ and $\lambda > 0$,

$$\mathbb{E}_{\mu,\nu}\left[\exp\left(\lambda\Big\{\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n) - \mathbb{E}_{\mu,\nu}[\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n)]\Big\}\right)\right] \leq \exp(\lambda^2\varphi_{\mu,\nu,p}n^{-1}). \tag{7.15}$$

Let $\rho_0 \in \mathcal{P}(\mathbb{S}^{d-1})$. By taking the expectation of Equation 7.15 with respect to ρ_0, then using Fubini's theorem to interchange the expectation over ρ_0 and the one over μ, ν, we obtain

$$\mathbb{E}_{\mu,\nu}\mathbb{E}_{\boldsymbol{\theta}\sim\rho_0}\left[\exp\left(\lambda\{\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n) - \mathbb{E}_{\mu,\nu}[\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n)]\}\right)\right] \leq \exp(\lambda^2\varphi_{\mu,\nu,p}n^{-1}). \tag{7.16}$$

By definition of the Wasserstein distance between empirical, univariate distributions Equation 7.2, one can prove that $\boldsymbol{\theta} \mapsto \lambda\{\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n) - \mathbb{E}_{\mu,\nu}[\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n)]\}$ is a bounded real-valued function on $\mathbb{S}^{d-1}$. Therefore, we can apply LemmA7.7.1 to rewrite Equation 7.16 as follows.

$$\begin{aligned}\mathbb{E}_{\mu,\nu}\Big[\exp\Big(\sup_{\rho\in\mathcal{P}(\Theta)}\big[\mathbb{E}_{\boldsymbol{\theta}\sim\rho}\big[\lambda\{\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n) - \mathbb{E}_{\mu,\nu}[\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n)]\}\big] - \mathrm{KL}(\rho||\rho_0)\big]\Big)\Big] \\ \leq \exp(\lambda^2\varphi_{\mu,\nu,p}n^{-1}),\end{aligned} \tag{7.17}$$

which, using the linearity of the expectation along with the definition of SW Equation 7.3, is equivalent to

$$\mathbb{E}_{\mu,\nu}\Big[\exp\Big(\sup_{\rho\in\mathcal{P}(\Theta)}\big[\lambda\{\mathrm{SW}_p^p(\mu_n, \nu_n; \rho) - \mathbb{E}_{\mu,\nu}[\mathrm{SW}_p^p(\mu_n, \nu_n; \rho)]\} - \mathrm{KL}(\rho||\rho_0)\big]\Big)\Big] \leq \exp(\lambda^2\varphi_{\mu,\nu,p}n^{-1}),$$

or,

$$\mathbb{E}_{\mu,\nu}\Big[\exp\Big(\sup_{\rho\in\mathcal{P}(\Theta)}\big[\lambda\{\mathrm{SW}_p^p(\mu_n, \nu_n; \rho) - \mathbb{E}_{\mu,\nu}[\mathrm{SW}_p^p(\mu_n, \nu_n; \rho)]\} - \mathrm{KL}(\rho||\rho_0)\big] - \lambda^2\varphi_{\mu,\nu,p}n^{-1}\Big)\Big] \leq 1. \tag{7.18}$$

Let $s > 0$. By the Chernoff bound $\Big(\mathbb{P}(X > a) = \mathbb{P}(e^{s.X} \geq e^{s.a}) \leq \mathbb{E}[e^{t.X}]e^{-t.a}\Big)$

$$\begin{aligned}
&\mathbb{P}_{\mu,\nu}\Big(\sup_{\rho\in\mathcal{P}(\Theta)}\big[\lambda\big\{\mathrm{SW}_p^p(\mu_n,\nu_n;\rho) - \mathbb{E}_{\mu,\nu}[\mathrm{SW}_p^p(\mu_n,\nu_n;\rho)]\big\} - \mathrm{KL}(\rho||\rho_0)\big] - \lambda^2\varphi_{\mu,\nu,p}n^{-1} > s\Big)\\
&\leq \mathbb{E}_{\mu,\nu}\Big[\exp\Big(\sup_{\rho\in\mathcal{P}(\Theta)}\big[\lambda\big\{\mathrm{SW}_p^p(\mu_n,\nu_n;\rho) - \mathbb{E}_{\mu,\nu}[\mathrm{SW}_p^p(\mu_n,\nu_n;\rho)]\big\} - \mathrm{KL}(\rho||\rho_0)\big] - \lambda^2\varphi_{\mu,\nu,p}n^{-1}\Big)\Big]\exp(-s)\\
&\leq 1\cdot\exp(-s) = \exp(-s),
\end{aligned}$$

where the last inequality follows from Equation 7.18.
Let $e^{-s} = \varepsilon$ such that $s = \log(1/\varepsilon)$. Then,

$$\mathbb{P}_{\mu,\nu}\Big(\exists\rho\in\mathcal{P}(\mathbb{S}^{d-1}),\ \lambda\big\{\mathrm{SW}_p^p(\mu_n,\nu_n;\rho) - \mathbb{E}_{\mu,\nu}[\mathrm{SW}_p^p(\mu_n,\nu_n;\rho)]\big\} - \mathrm{KL}(\rho||\rho_0) - \lambda^2\varphi_{\mu,\nu,p}n^{-1} > \log(1/\varepsilon)\Big) \leq \varepsilon. \tag{7.19}$$

Taking the complement of Equation 7.19 and rearranging the terms yields

$$\begin{aligned}
&\mathbb{P}_{\mu,\nu}\Big(\forall\rho\in\mathcal{P}(\mathbb{S}^{d-1}),\ \mathrm{SW}_p^p(\mu_n,\nu_n;\rho) < \mathbb{E}_{\mu,\nu}[\mathrm{SW}_p^p(\mu_n,\nu_n;\rho)] + \lambda^{-1}\big\{\mathrm{KL}(\rho||\rho_0) + \log(1/\varepsilon)\big\} + \lambda\varphi_{\mu,\nu,p}n^{-1}\Big)\\
&\quad\geq 1-\varepsilon.
\end{aligned} \tag{7.20}$$

Our final bound results from assuming there exists $\psi_{\mu,\nu,p}(n)$ such that,

$$\mathbb{E}_{\mu,\nu}|\mathrm{SW}_p^p(\mu_n,\nu_n;\rho) - \mathrm{SW}_p^p(\mu,\nu;\rho)| \leq \psi_{\mu,\nu,p}(n). \tag{7.21}$$

□

7.7.2 Proof of Proposition 7.3.4

To prove Proposition 7.3.4, we leverage a concentration result that appears in the proof of McDiarmid's inequality (recalled in Theorem 7.7.3), and which relies on the *bounded differences property* (Definition 7.7.2).

Definition 7.7.2 (Bounded differences property)**.** *Let $\mathsf{X}\subset\mathbb{R}$, $n\in\mathbb{N}^*$ and $c = \{c_i\}_{i=1}^n \in \mathbb{R}^n$. A mapping $f : \mathsf{X}^n \to \mathbb{R}$ is said to satisfy the* c-bounded differences property *if for $i\in\{1,\dots,n\}$, $\{x_i\}_{i=1}^n \in \mathsf{X}^n$ and $x_i'\in\mathsf{X}$,*

$$|f(x_1,\dots,x_n) - f(x_1,\dots,x_{i-1},x_i',x_{i+1},\dots,x_n)| \leq c_i. \tag{7.22}$$

Theorem 7.7.3 ([McDiarmid, 1989])**.** *Let $(x_i)_{i=1}^n$ be a sequence of $n\in\mathbb{N}^*$ independent random variables with x_i valued in $\mathsf{X}\subset\mathbb{R}$ for $i\in\{1,\dots,n\}$. Let $\boldsymbol{c} = \{c_i\}_{i=1}^n \in \mathbb{R}^n$ and $f : \mathsf{X}^n \to \mathbb{R}$ satisfying the c-bounded differences property. Then, for any $\lambda > 0$,*

$$\mathbb{E}\big[\exp(\lambda\{f - \mathbb{E}[f]\})\big] \leq \exp(\lambda^2\|\boldsymbol{c}\|^2/8). \tag{7.23}$$

The proof of Proposition 7.3.4 consists in applying Theorem 7.7.3 to the mapping $(\boldsymbol{x}_1,\dots,\boldsymbol{x}_n, \boldsymbol{y}_1,\dots,\boldsymbol{y}_n) \mapsto \mathrm{W}_p^p(\theta_\sharp^*\mu,\theta_\sharp^*\nu)$, for any $\boldsymbol{\theta}\in\mathbb{S}^{d-1}$. To this end, we show that the one-dimensional Wasserstein distance satisfies the bounded differences property, assuming bounded supports.

Lemma 7.7.4. *Let $\mathsf{X} \subset \mathbb{R}$ be a bounded set with diameter $\Delta = \sup_{(x,x')\in\mathsf{X}^2} \|x - x'\| < +\infty$. Then, the mapping $f : \mathbb{R}^{2n} \to \mathbb{R}_+$ defined for $\{x_i\}_{i=1}^n, \{y_i\}_{i=1}^n \in \mathbb{R}^n$ as*

$$f(x_1, \dots, x_n, y_1, \dots, y_n) = W_p^p(\mu_n, \nu_n) \tag{7.24}$$

satisfies the c-bounded differences property with $c_i = \Delta^p/n$ for $i \in \{1, \dots, n\}$.

Proof. For clarity purposes, we start by introducing some notations. Let $n \in \mathbb{N}^*$ and denote by μ_n, ν_n the empirical distributions supported over $\{x_i\}_{i=1}^n, \{y_i\}_{i=1}^n \in \mathbb{R}^n$ respectively. For $i \in \{1, \dots, n\}$, let $x_i' \in \mathbb{R}$ and μ_n' the empirical distribution supported on $(x_1, \dots, x_{i-1}, x_i', x_{i+1}, \dots, x_n) \in \mathbb{R}^n$. Let $\sigma : \{1, \dots, n\} \to \{1, \dots, n\}$ (respectively, $\sigma' : \{1, \dots, n\} \to \{1, \dots, n\}$) s.t. for $i \in \{1, \dots, n\}$, $x_{\sigma(i)}$ (resp., $x'_{\sigma'(i)}$) is the i-th smallest value of $\{x_i\}_{i=1}^n$ (resp., $(x_1, \dots, x_{i-1}, x_i', x_{i+1}, \dots, x_n)$). By definition of the Wasserstein distance between univariate distributions Equation 7.2,

$$\begin{aligned}
\mathrm{W}_p^p(\mu_n, \nu_n) - \mathrm{W}_p^p(\mu_n', \nu_n) &= \frac{1}{n}\sum_{i=1}^n |x_{\sigma(i)} - y_{(i)}|^p - \frac{1}{n}\sum_{i=1}^n |x'_{\sigma'(i)} - y_{(i)}|^p && (7.25)\\
&\leq \frac{1}{n}\sum_{i=1}^n |x_{\sigma'(i)} - y_{(i)}|^p - \frac{1}{n}\sum_{i=1}^n |x'_{\sigma'(i)} - y_{(i)}|^p && (7.26)\\
&\leq \frac{1}{n}\left(|x_{\sigma'(i)} - y_{(i)}|^p - |x'_{\sigma'(i)} - y_{(i)}|^p\right) && (7.27)\\
&\leq \frac{\Delta^p}{n}. && (7.28)
\end{aligned}$$

We can use the same arguments to prove that $\mathrm{W}_p^p(\mu_n', \nu_n) - \mathrm{W}_p^p(\mu_n, \nu_n) \leq \Delta^p/n$, hence

$$\left|\mathrm{W}_p^p(\mu_n, \nu_n) - \mathrm{W}_p^p(\mu_n', \nu_n)\right| \leq \frac{\Delta^p}{n}. \tag{7.29}$$

On the other hand, let $y_i' \in \mathbb{R}$ and ν_n' the empirical distribution over $(y_1, \dots, y_{i-1}, y_i', y_{i+1}, \dots, y_n)$. Since the Wasserstein distance is symmetric,

$$\left|\mathrm{W}_p^p(\mu_n, \nu_n) - \mathrm{W}_p^p(\mu_n, \nu_n')\right| = \left|\mathrm{W}_p^p(\nu_n, \mu_n) - \mathrm{W}_p^p(\nu_n', \mu_n)\right| \leq \frac{\Delta^p}{n}, \tag{7.30}$$

where the last inequality results from Equation 7.29.

□

Remark 7.7.5. *LemmA7.7.4 is a particular case of [Weed, 2019, Proposition 20], which establishes a concentration bound for $W_p^p(\mu, \mu_n)$ around its expectation on any finite-dimensional compact space by exploiting McDiarmid's inequality along with the Kantorovich duality. We thus use similar arguments to prove Proposition 7.3.4, except we leverage the closed-form expression of one-dimensional Wasserstein distances instead of the dual formulation since we compare univariate projected distributions. The corresponding proof was detailed for completeness.*

Proof of Proposition 7.3.4. Let $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$. Assuming A1 implies that $\theta_\sharp^*\mu, \theta_\sharp^*\nu$ are both supported on a bounded domain, whose diameter is denoted by $\Delta_{\boldsymbol{\theta}}$ and satisfies $\Delta_{\boldsymbol{\theta}} < \Delta$. Hence, by LemmA7.7.4, $f_{\boldsymbol{\theta}} : (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \boldsymbol{y}_1, \dots, \boldsymbol{y}_n) \mapsto \mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n)$ satisfies the bounded differences

property. We can then apply Theorem 7.7.3 to bound the moment-generating function of $f_{\boldsymbol{\theta}} - \mathbb{E}f_{\boldsymbol{\theta}}$ as follows.

$$\mathbb{E}\big[\exp(\lambda\{f - \mathbb{E}[f]\})\big] \leq \exp(\lambda^2 \sum_{i=1}^{2n} (\Delta_{\boldsymbol{\theta}}^p/n)^2/8) \tag{7.31}$$

$$\leq \exp(\lambda^2 \Delta_{\boldsymbol{\theta}}^{2p}/(4n)) \leq \exp(\lambda^2 \Delta^{2p}/(4n)), \tag{7.32}$$

which concludes the proof. □

7.7.3 Proof of Proposition 7.3.5

Recent work have bounded $\mathbb{E}|\mathrm{SW}_p(\mu_n, \nu_n; \rho) - \mathrm{SW}_p(\mu, \nu; \rho)|$ or $\mathbb{E}|\mathrm{SW}_p(\mu, \mu_n; \rho)|$ for specific choices of $\rho \in \mathcal{P}(\mathbb{S}^{d-1})$ [Nadjahi, 2020b; Manole, 2020; Nguyen, 2021; Lin, 2021]. These results do not exactly correspond to what Theorem 7.3.1 requires, *i.e.* a bound on $\mathbb{E}|\mathrm{SW}_p^p(\mu_n, \nu_n; \rho) - \mathrm{SW}_p^p(\mu, \nu; \rho)|$. We bound the latter quantity in Proposition 7.3.5, by specifying the proof techniques in [Manole, 2020] under A1, then generalizing a result in [Nadjahi, 2020b].

Lemma 7.7.6. *Let $p \in [1, +\infty)$ and $\mathsf{X} \subset \mathbb{R}$ a bounded set, with diameter denoted by $\Delta < +\infty$. Let $\mu, \nu \in \mathcal{P}(\mathsf{X})$ and denote by μ_n, ν_n the empirical distributions supported over $n \in \mathbb{N}^*$ samples i.i.d. from μ, ν respectively. Then,*

$$\mathbb{E}\big|\mathrm{W}_p^p(\mu_n, \nu_n) - \mathrm{W}_p^p(\mu, \nu)\big| \leq p\Delta^{p-1}\{J_1(\mu) + J_1(\nu)\}n^{-1/2}, \tag{7.33}$$

where for $\xi \in \{\mu, \nu\}$,

$$J_1(\xi) = \int_{-\infty}^{+\infty} \sqrt{F_\xi(x)(1 - F_\xi(x))}\mathrm{d}x. \tag{7.34}$$

Proof. Lemma 7.7.6 is obtained by adapting the techniques used in the proof of [Manole, 2020, Lemma B.3]. We provide the detailed proof for completeness.
Starting from the definition of $\mathrm{W}_p^p(\mu_n, \nu_n)$ Equation 7.2, then using a Taylor expansion of $(x, y) \mapsto |x - y|^p$ around $(x, y) = (F_\mu^{-1}(t), F_\nu^{-1}(t))$, we obtain

$$\begin{aligned}
\mathrm{W}_p^p(\mu_n, \nu_n) &= \int_0^1 \big|F_{\mu_n}^{-1}(t) - F_{\nu_n}^{-1}(t)\big|^p \mathrm{d}t \\
&= \int_0^1 \big|F_{\mu}^{-1}(t) - F_{\nu}^{-1}(t)\big|^p \mathrm{d}t \qquad (7.35) \\
&+ \int_0^1 p\,\mathrm{sgn}\big(\tilde{F}_{\mu_n}^{-1}(t) - \tilde{F}_{\nu_n}^{-1}(t)\big)\big|\tilde{F}_{\mu_n}^{-1}(t) - \tilde{F}_{\nu_n}^{-1}(t)\big|^{p-1}\big\{(F_{\mu_n}^{-1}(t) - F_{\mu}^{-1}(t)) - (F_{\nu_n}^{-1}(t) - F_{\nu}^{-1}(t))\big\}\mathrm{d}t, \qquad (7.36)
\end{aligned}$$

where $\mathrm{sgn}(\cdot)$ denotes the sign function, $\tilde{F}_{\mu_n}^{-1}(t)$ a real number between $F_{\mu_n}^{-1}(t)$ and $F_{\mu}^{-1}(t)$, and $\tilde{F}_{\nu_n}^{-1}(t)$ a real number between $F_{\nu_n}^{-1}(t)$ and $F_{\nu}^{-1}(t)$.

By definition, Equation 7.35 is exactly $\mathrm{W}_p^p(\mu,\nu)$, so we obtain

$$
\begin{aligned}
&|\mathrm{W}_p^p(\mu_n,\nu_n) - \mathrm{W}_p^p(\mu,\nu)| \\
&= \Big| \int_0^1 p\,\mathrm{sgn}\big(\tilde{F}_{\mu_n}^{-1}(t) - \tilde{F}_{\nu_n}^{-1}(t)\big)\big|\tilde{F}_{\mu_n}^{-1}(t) - \tilde{F}_{\nu_n}^{-1}(t)\big|^{p-1}\big\{(F_{\mu_n}^{-1}(t) - F_{\mu}^{-1}(t)) - (F_{\nu_n}^{-1}(t) - F_{\nu}^{-1}(t))\big\}\mathrm{d}t\Big| \\
&\leq p\int_0^1 \big|\tilde{F}_{\mu_n}^{-1}(t) - \tilde{F}_{\nu_n}^{-1}(t)\big|^{p-1}\Big\{\big|F_{\mu_n}^{-1}(t) - F_{\mu}^{-1}(t)\big| + \big|F_{\nu_n}^{-1}(t) - F_{\nu}^{-1}(t)\big|\Big\}\mathrm{d}t \qquad (7.37) \\
&\leq p \sup_{t\in(0,1)} \big|\tilde{F}_{\mu_n}^{-1}(t) - \tilde{F}_{\nu_n}^{-1}(t)\big|^{p-1}\Big\{\mathrm{W}_1(\mu_n,\mu) + \mathrm{W}_1(\nu_n,\nu)\Big\}, \qquad (7.38)
\end{aligned}
$$

where Equation 7.37 follows from the triangle inequality and Equation 7.38 results from the definition of the Wasserstein distance of order 1 between univariate distributions.
We then bound $\sup_{t\in(0,1)} \big|\tilde{F}_{\mu_n}^{-1}(t) - \tilde{F}_{\nu_n}^{-1}(t)\big|^{p-1}$ from above. By the definition of $\tilde{F}_{\mu_n}^{-1}(t), \tilde{F}_{\nu_n}^{-1}(t)$ for $t \in (0,1)$, we distinguish the following four cases :

i) $\tilde{F}_{\mu_n}^{-1}(t) \leq F_{\mu_n}^{-1}(t)$, $\tilde{F}_{\nu_n}^{-1}(t) \leq F_{\nu_n}^{-1}(t)$,
ii) $\tilde{F}_{\mu_n}^{-1}(t) \leq F_{\mu}^{-1}(t)$, $\tilde{F}_{\nu_n}^{-1}(t) \leq F_{\nu}^{-1}(t)$,
iii) $\tilde{F}_{\mu_n}^{-1}(t) \leq F_{\mu_n}^{-1}(t)$, $\tilde{F}_{\nu_n}^{-1}(t) \leq F_{\nu}^{-1}(t)$,
iv) $\tilde{F}_{\mu_n}^{-1}(t) \leq F_{\mu}^{-1}(t)$, $\tilde{F}_{\nu_n}^{-1}(t) \leq F_{\nu_n}^{-1}(t)$.

Hence, using the definition of quantile functions and the fact that the supports of μ,ν are assumed to be bounded, we obtain

$$
\sup_{t\in(0,1)} \big|\tilde{F}_{\mu_n}^{-1}(t) - \tilde{F}_{\nu_n}^{-1}(t)\big|^{p-1} \leq \Delta^{p-1}. \qquad (7.39)
$$

We conclude that,

$$
|\mathrm{W}_p^p(\mu_n,\nu_n) - \mathrm{W}_p^p(\mu,\nu)| \leq p\Delta^{p-1}\Big\{\mathrm{W}_1(\mu_n,\mu) + \mathrm{W}_1(\nu_n,\nu)\Big\}. \qquad (7.40)
$$

Our final result follows from [Bobkov, 2019, Theorem 3.2], which gives us

$$
\mathbb{E}[\mathrm{W}_1(\mu_n,\mu)] \leq J_1(\mu)n^{-1/2}, \;\; \mathbb{E}[\mathrm{W}_1(\nu_n,\nu)] \leq J_1(\nu)n^{-1/2}. \qquad (7.41)
$$

Note that since μ,ν are supported on a bounded domain, the moment of μ (or ν) of order $k \in \mathbb{N}^*$ is finite, which implies that $J_1(\mu), J_1(\nu)$ are both finite [Bobkov, 2019, Section 3.1].

□

Proof of Proposition 7.3.5. Let $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$. Under A1, one can easily prove that $\theta_\sharp^*\mu, \theta_\sharp^*\nu$ are supported on a bounded domain with diameter $\Delta_{\boldsymbol{\theta}} \leq \Delta < +\infty$. Therefore, by LemmA7.7.6,

$$
\mathbb{E}\big|\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n) - \mathrm{W}_p^p(\theta_\sharp^*\mu, \theta_\sharp^*\nu)\big| \leq p\Delta^{p-1}\big\{J_1(\theta_\sharp^*\mu) + J_1(\theta_\sharp^*\nu)\big\}n^{-1/2}. \qquad (7.42)
$$

Next, we adapt the proof techniques in [Nadjahi, 2020b, Theorem 4] to establish the following inequality : for *any* $\rho \in \mathcal{P}(\mathbb{S}^{d-1})$,

$$
\mathbb{E}|\mathrm{SW}_p^p(\mu_n,\nu_n;\rho) - \mathrm{SW}_p^p(\mu,\nu;\rho)| \leq \int_{\mathbb{S}^{d-1}} \mathbb{E}|\mathrm{W}_p^p(\theta_\sharp^*\mu_n, \theta_\sharp^*\nu_n) - \mathrm{W}_p^p(\theta_\sharp^*\mu, \theta_\sharp^*\nu)|\mathrm{d}\rho(\boldsymbol{\theta}). \qquad (7.43)
$$

Hence, by plugging Equation 7.42 in Equation 7.43, we conclude that

$$\mathbb{E}|\mathrm{SW}_p^p(\mu_n,\nu_n;\rho)-\mathrm{SW}_p^p(\mu,\nu;\rho)| \leq p\Delta^{p-1}\left(\int_{\mathbb{S}^{d-1}}\{J_1(\theta_\sharp^*\mu)+J_1(\theta_\sharp^*\nu)\}\mathrm{d}\rho(\boldsymbol{\theta})\right)n^{-1/2}. \quad (7.44)$$

□

7.7.4 Final bound for bounded supports

By incorporating Propositions 7.3.4 and 7.3.5 in Theorem 7.3.1, we obtain the following result. Corollary 7.7.7 corresponds to a specialization of our generic bound when considering distributions with bounded supports.

Corollary 7.7.7. *Let $p \in [1,+\infty)$ and assume A1. Let $\rho_0 \in \mathcal{P}(\mathbb{S}^{d-1})$ and $\delta > 0$. Then, with probability at least $1-\delta$, for all $\rho \in \mathcal{P}(\mathbb{S}^{d-1})$ and $\lambda > 0$,*

$$\begin{aligned}\mathrm{SW}_p^p(\mu_n,\nu_n;\,\rho) \leq \mathrm{SW}_p^p(\mu,\nu;\,\rho) &+ \{KL(\rho||\rho_0)+\log(1/\delta)\}\,\lambda^{-1} \\ &+ \lambda\Delta^{2p}(4n)^{-1} + p\Delta^{p-1}\{SJ(\mu)+SJ(\nu)\}n^{-1/2}. \end{aligned} \quad (7.45)$$

7.7.5 Proof of Proposition 7.3.7

When the supports of the distributions are not bounded, LemmA7.7.4 does not hold true, thus preventing the use of McDiarmid's inequality. Hence, to compute $\varphi_{\mu,\nu,p}$, we may use extensions of McDiarmid's inequality which replace the finite-diameter constraint by conditions on the moments of the distributions.
In particular, Proposition 7.3.7 follows from applying [Kontorovich, 2014, Theorem 1], a concentration result based on the notion of *sub-Gaussian diameter*.

Definition 7.7.8 (Sub-Gaussian diameter [Kontorovich, 2014])**.** *Let η be a distance function and (X,η,μ) be the associated metric probability space. Consider a sequence of $n \in \mathbb{N}^*$ independent random variables $(x_i)_{i=1}^n$ with x_i distributed from μ for $i \in \{1,\dots,n\}$. Let $\Xi(\mathsf{X})$ be the random variable defined by*

$$\Xi(\mathsf{X}) = \varepsilon\eta(X,x'), \quad (7.46)$$

where X, x' are two independent realizations from μ and ε is a random variable valued in $\{-1,1\}$ s.t. $p(\varepsilon=1)=1/2$ and ε is independent from X, x'.
Additionally, suppose there exists $\sigma > 0$ s.t. for $\lambda \in \mathbb{R}$, $\mathbb{E}_\mu[\exp(\lambda X)] \leq \exp(\sigma^2\lambda^2/2)$. The sub-Gaussian diameter *of (X,η,μ), denoted by $\Delta_{SG}(\mathsf{X})$, is defined as*

$$\Delta_{SG}(\mathsf{X}) = \sigma(\Xi(\mathsf{X})). \quad (7.47)$$

Note that $\Delta_{\mathrm{SG}} \leq \Delta$ [Kontorovich, 2014, Lemma 1], and a set with infinite diameter may have a finite sub-Gaussian diameter. Hence, Theorem 7.7.9 relaxes the conditions of Theorem 7.7.3.

Theorem 7.7.9 (Theorem 1 [Kontorovich, 2014])**.** *Let $(\mathsf{X},\|\cdot\|_1,\mu)$ be a metric probability space with $\mathsf{X} \subset \mathbb{R}^d$ and $\|\cdot\|_1$ the L_1-norm. Consider a sequence of random variables $(\boldsymbol{x}_i)_{i=1}^n$ i.i.d. from*

μ. Let $f : \mathsf{X}^n \to \mathbb{R}$ s.t. f is 1-Lipschitz, i.e. for any $(\boldsymbol{x}, \boldsymbol{x}') \in \mathsf{X}^n \times \mathsf{X}^n$, $|f(\boldsymbol{x}) - f(\boldsymbol{x}')| \leq \|\boldsymbol{x} - \boldsymbol{x}'\|_1$. Then, $\mathbb{E}[f] < +\infty$ and for $\lambda > 0$,

$$\mathbb{E}\left[\exp(\lambda\{f - \mathbb{E}[f]\})\right] \leq \exp(\lambda^2 n \Delta_{SG}(\mathsf{X})^2/2). \tag{7.48}$$

As discussed in [Kontorovich, 2014], the sub-Gaussian distributions on $\mathbb{R}$ are precisely those for which $\Delta_{\mathrm{SG}}(\mathbb{R}) < +\infty$. This allows the application of Theorem 7.7.9 under A2, which yields Proposition 7.3.7.

Proof of Proposition 7.3.7. First, the Wasserstein distance between discrete, univariate distributions is $1/n$-Lipschitz. Indeed, consider μ'_n, ν'_n supported over $\{x'_i\}_{i=1}^n, \{y'_i\}_{i=1}^n \in \mathbb{R}^n$; then, by definition,

$$\mathrm{W}_1(\mu'_n, \nu'_n) = n^{-1} \sum_{i=1}^n |x'_{(i)} - y'_{(i)}| \leq n^{-1} \sum_{i=1}^n |x'_i - y'_i|. \tag{7.49}$$

Let $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$. Since $\mu, \nu \in \mathcal{P}(\mathbb{R}^d)$ are assumed to be sub-Gaussian (by A2), then $\theta^*_\sharp \mu, \theta^*_\sharp \nu$ are sub-Gaussian distributions with respective variance proxies σ^2, τ^2 (Definition 7.3.6). Besides, there exists almost surely $\hat{\sigma}^2, \hat{\tau}^2$ s.t. $\theta^*_\sharp \mu_n, \theta^*_\sharp \nu_n$ are sub-Gaussian [Mena, 2019, Lemma A.2].
Consider the metric probability space $(\mathbb{R}, \|\cdot\|_1, \xi_n)$, with $\xi_n \in \{\mu_n, \nu_n\}$. By Definition 7.7.8 and the properties of the sum of two sub-Gaussian distributions, $\Delta_{\mathrm{SG}}(\mathbb{R}) = \sqrt{2}\hat{\sigma}$ if $\xi_n = \mu_n$, and $\Delta_{\mathrm{SG}}(\mathbb{R}) = \sqrt{2}\hat{\tau}$ if $\xi_n = \nu$.
Finally, let $\lambda > 0$. By applying Theorem 7.7.9 to $f_{\boldsymbol{\theta}} : \mathsf{X}^{2n} \to \mathbb{R}_+$ defined for $\{x_i\}_{i=1}^n, \{y_i\}_{i=1}^n \in \mathbb{R}^n$ by $f_{\boldsymbol{\theta}}(x_1, \ldots, x_n, y_1, \ldots, y_n) = n\mathrm{W}_1(\theta^*_\sharp \mu_n, \theta^*_\sharp \nu_n)$, which is 1-Lipschitz by Equation 7.49, we obtain

$$\mathbb{E}\big[\exp(\lambda n^{-1}\{f_{\boldsymbol{\theta}} - \mathbb{E}[f_{\boldsymbol{\theta}}]\})\big] \leq \exp\left(\lambda^2 n^{-1}(\hat{\sigma}^2 + \hat{\tau}^2)\right), \tag{7.50}$$

which concludes the proof.

□

7.7.6 Proof of Proposition 7.3.9

Proposition 7.3.9 results from the same arguments as in the proof of [Lei, 2020, Corollary 5.2]. The latter result is obtained by applying a generalized McDiarmid's inequality, which we recall in Theorem 7.7.10.

Theorem 7.7.10 (Bernstein-type McDiarmid's inequality [Lei, 2020])**.** *Let $\mathsf{X} \subset \mathbb{R}^d$ and $X = (\boldsymbol{x}_i)_{i=1}^n$ be a sequence of $n \in \mathbb{N}^*$ random variables i.i.d. from $\mu \in \mathcal{P}(\mathsf{X})$. Let $f : \mathsf{X}^n \to \mathbb{R}$ s.t. $\mathbb{E}|f| < \infty$. For $i \in \{1, \ldots, n\}$, let $\boldsymbol{x}'_i$ be an independent copy of $\boldsymbol{x}_i$ and $X'_{(i)} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}'_i, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_n)$. Assume that for $i \in \{1, \ldots, n\}$, there exists $c_i, M > 0$ such that for $k \geq 2$,*

$$\mathbb{E}[f(X) - f(X'_{(i)}) \mid X_{-i}] \leq c_i^2 k! M^{k-2}/2, \tag{7.51}$$

where $X_{-i} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \dots, \boldsymbol{x}_n)$. *Then, for* $\lambda > 0$ *s.t.* $\lambda M < 1$,

$$\mathbb{E}\big[\exp\{\lambda(f - \mathbb{E}[f])\}\big] \leq \exp\Big(\lambda^2 \|c\|^2 / \{2(1-\lambda M)\}\Big). \tag{7.52}$$

Proof of Proposition 7.3.9. Let $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$. For $i \in \{1, \dots, n\}$, let $\boldsymbol{x}_i' \in \mathsf{X} \subset \mathbb{R}^d$ and μ_n' the empirical distribution supported on $(\boldsymbol{x}_1, \dots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_i', \boldsymbol{x}_{i+1}, \dots, \boldsymbol{x}_n) \in \mathsf{X}^n$. Since W_1 satisfies the triangle inequality,

$$\begin{aligned} |\mathrm{W}_1(\theta_\sharp^* \mu_n, \theta_\sharp^* \nu_n) - \mathrm{W}_1(\theta_\sharp^* \mu_n', \theta_\sharp^* \nu_n)| &\leq \mathrm{W}_1(\theta_\sharp^* \mu_n, \theta_\sharp^* \mu_n') && (7.53)\\ &\leq n^{-1} \|\boldsymbol{\theta}^\top (\boldsymbol{x}_i - \boldsymbol{x}_i')\| && (7.54)\\ &\leq n^{-1} \|\boldsymbol{x}_i - \boldsymbol{x}_i'\|, && (7.55) \end{aligned}$$

where the last inequality follows from the Cauchy-Schwarz inequality and $\|\boldsymbol{\theta}\| = 1$.
Similarly, for $i \in \{1, \dots, n\}$, let $\boldsymbol{y}_i' \in \mathsf{X} \subset \mathbb{R}^d$ and denote ν_n' the empirical distribution supported on $(\boldsymbol{y}_1, \dots, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i', \boldsymbol{y}_{i+1}, \dots, \boldsymbol{y}_n) \in \mathsf{X}^n$. By symmetry of W_1 and Equation 7.55, we have

$$\begin{aligned} |\mathrm{W}_1(\theta_\sharp^* \mu_n, \theta_\sharp^* \nu_n) - \mathrm{W}_1(\theta_\sharp^* \mu_n, \theta_\sharp^* \nu_n')| &= |\mathrm{W}_1(\theta_\sharp^* \nu_n, \theta_\sharp^* \mu_n) - \mathrm{W}_1(\theta_\sharp^* \nu_n', \theta_\sharp^* \mu_n)| && (7.56)\\ &\leq n^{-1} \|\boldsymbol{y}_i - \boldsymbol{y}_i'\|. && (7.57) \end{aligned}$$

We deduce from Equation 7.55, 7.57 and A3 that the mapping $f_{\boldsymbol{\theta}} : \mathsf{X}^{2n} \to \mathbb{R}_+$ defined by $f_{\boldsymbol{\theta}}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \boldsymbol{y}_1, \dots, \boldsymbol{y}_n) = \mathrm{W}_1(\theta_\sharp^* \mu_n, \theta_\sharp^* \nu_n)$ satisfies the $(\sigma_\star^2, b_\star)$-Bernstein condition, with $\sigma_\star^2 = \max(\sigma^2, \tau^2)$, $b_\star = \max(b, c)$.
A direct application of Theorem 7.7.10 to $f_{\boldsymbol{\theta}}$ gives the final result, *i.e.*

$$\mathbb{E}\Big[\exp\Big(\lambda\{\mathrm{W}_1(\theta_\sharp^* \mu_n, \theta_\sharp^* \nu_n) - \mathbb{E}[\mathrm{W}_1(\theta_\sharp^* \mu_n, \theta_\sharp^* \nu_n)]\}\Big)\Big] \leq \exp\Big(2\sigma_\star^2 \lambda^2 n^{-2} (1 - 2b_\star \lambda n^{-1})^{-1}\Big). \tag{7.58}$$

□

7.7.7 Final bound for unbounded supports

Before deriving the specialization of Theorem 7.3.1 for distributions with unbounded supports, we recall a useful bound on $\mathrm{SW}_p^p(,;\pi)$ with $\pi = \mathcal{U}(\mathbb{S}^{d-1})$ (Theorem 7.7.11), which can be generalized for SW based on any $\rho \in \mathcal{P}(\mathbb{S}^{d-1})$ by adapting the proof techniques in [Manole, 2020].

Theorem 7.7.11 ([Manole, 2020])**.** *Let* $p \geq 1$, $q > 2p$, $s \geq 1$ *and* $\pi = \mathcal{U}(\mathbb{S}^{d-1})$. *Denote* $\mathcal{P}_{p,q}(s) = \Big\{\mu \in \mathcal{P}(\mathbb{R}^d) \ : \ \int_{\mathbb{S}^{d-1}} \mathbb{E}_\mu[|\boldsymbol{\theta}^\top \boldsymbol{x}|^q]^{p/q} \mathrm{d}\pi(\boldsymbol{\theta}) \leq s\Big\}$. *Let* $\mu, \nu \in \mathcal{P}_{p,q}(s)$. *Then, there exists a constant* $C(p,q) > 0$ *depending on* p, q *such that,*

$$\mathbb{E}|\mathrm{SW}_p^p(\mu_n, \nu_n; \pi) - \mathrm{SW}_p^p(\mu, \nu; \pi)| \leq C(p,q) s \log(n)^{1/2} n^{-1/2}. \tag{7.59}$$

We show that under A2 or A3, the assumptions in Theorem 7.7.11 are satisfied, thus allowing its application in these two settings. This yields Corollaries 7.7.12 and 7.7.13, which we state and prove hereafter.

Corollary 7.7.12. *Assume A2 and let $\rho \in \mathcal{P}(\mathbb{S}^{d-1})$. Then, there exists $C'(p) > 0$ such that,*

$$\mathbb{E}|\mathrm{SW}_p^p(\mu_n, \nu_n; \rho) - \mathrm{SW}_p^p(\mu, \nu; \rho)| \leq C'(p)(4\sigma_\star^2)^p \log(n)^{1/2} n^{-1/2}. \tag{7.60}$$

Proof. Under A2, μ, ν are assumed to be sub-Gaussian, so the moments of $\theta_\sharp^* \mu, \theta_\sharp^* \nu$ can be bounded for any $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ as follows : for any $k \in \mathbb{N}^*$,

$$\mathbb{E}_\mu[|\boldsymbol{\theta}^\top \boldsymbol{x}|^{2k}] \leq k!(4\sigma^2)^k, \quad \mathbb{E}_\nu[|\boldsymbol{\theta}^\top \boldsymbol{y}|^{2k}] \leq k!(4\tau^2)^k. \tag{7.61}$$

We conclude that $\mu, \nu \in \mathcal{P}_{p,2(p+1)}(s)$ with $s = \{(p+1)!\}^{p/(2(p+1))}(4\sigma_\star^2)^p$ and $\sigma_\star^2 = \max(\sigma^2, \tau^2)$. The final result follows from applying Theorem 7.7.11. □

Corollary 7.7.13. *Assume A3 and let $\rho \in \mathcal{P}(\mathbb{S}^{d-1})$. Then, there exists $C'(p, q) > 0$ such that,*

$$\mathbb{E}|\mathrm{SW}_p^p(\mu_n, \nu_n;\, \rho) - \mathrm{SW}_p^p(\mu, \nu;\, \rho)| \leq C'(p, q)\sigma_\star^{2p/q} b_\star^{p(q-2)/q} \log(n)^{1/2} n^{-1/2}, \tag{7.62}$$

with $\sigma_\star^2 = \max(\sigma^2, \tau^2)$ and $b_\star = \max(b, c)$.

Proof. Under A3, the moments of μ, ν are bounded by the Bernstein condition. Therefore, using the definition of the push-forward measures along with the Cauchy-Scharz inequality, we obtain for any $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ and $k \in \mathbb{N}^*$,

$$\mathbb{E}_\mu[|\boldsymbol{\theta}^\top \boldsymbol{x}|^{2k}] \leq \sigma^2 k! b^{k-2}/2, \quad \mathbb{E}_\nu[|\boldsymbol{\theta}^\top \boldsymbol{y}|^{2k}] \leq \tau^2 k! c^{k-2}/2. \tag{7.63}$$

Let $q > 2p$. By Equation 7.63, $\mu, \nu \in \mathcal{P}_{p,q}(s)$ with $s = (\sigma_\star^2 q!/2)^{p/q} b_\star^{p(q-2)/q}$. The application of Theorem 7.7.11 concludes the proof.

□

We can finally provide the refined bounds under A2 and A3. On the one hand, incorporating Proposition 7.3.7 and Corollary 7.7.12 in Theorem 7.3.1 gives us the next corollary.

Corollary 7.7.14. *Assume A2. Let $\rho_0 \in \mathcal{P}(\mathbb{S}^{d-1})$ and $\delta > 0$. Then, with probability at least $1 - \delta$, for all $\rho \in \mathcal{P}(\mathbb{S}^{d-1})$ and $\lambda > 0$, there exists $C > 0$ such that*

$$\begin{aligned}\mathrm{SW}_1(\mu_n, \nu_n; \rho) \leq \mathrm{SW}_1(\mu, \nu; \rho) &+ \{KL(\rho||\rho_0) + \log(1/\delta)\}\lambda^{-1} \\ &+ \lambda(\hat{\sigma}^2 + \hat{\tau}^2)n^{-1} + C\max(\sigma^2, \tau^2)\log(n)^{1/2} n^{-1/2}.\end{aligned} \tag{7.64}$$

On the other hand, we leverage Proposition 7.3.9 and Corollary 7.7.13 to derive the specified bound below.

Corollary 7.7.15. *Assume A3 and denote $\sigma_\star^2 = \max(\sigma^2, \tau^2)$, $b_\star = \max(b, c)$. Let $\rho_0 \in \mathcal{P}(\mathbb{S}^{d-1})$ and $\delta > 0$. Then, with probability at least $1 - \delta$, for all $\rho \in \mathcal{P}(\mathbb{S}^{d-1})$ and $\lambda > 0$ s.t. $\lambda < (2b_\star)^{-1} n$,*

for $q > 2$, there exists $C(q) > 0$ such that

$$SW_1(\mu_n, \nu_n; \rho) \leq SW_1(\mu, \nu; \rho) + \{KL(\rho||\rho_0) + \log(1/\delta)\}\, \lambda^{-1} + 2\lambda\sigma_\star^2(1 - 2b_\star\lambda n^{-1})^{-1}n^{-2} + C(q)\sigma_\star^{2/q} b_\star^{(q-2)/q} \log(n)^{1/2} n^{-1/2}. \tag{7.65}$$

7.8 Appendix : additional details on our experiments

All our numerical experiments presented in Section 7.5 can be reproduced using the code we provide in https://github.com/rubenohana/PAC-Bayesian_Sliced-Wasserstein.git.

7.8.1 Details on the algorithmic procedure

For clarity purposes, we specify Algorithm 4 in the case where the optimization is performed over the space of von Mises-Fisher distributions (Definition 7.4.1). The corresponding procedure is given in Algorithm 5.

Algorithm 5 PAC-Bayes bound optimization for vMF-based SW

Input :

Datasets : $\boldsymbol{x}_{1:n} = (\boldsymbol{x}_i)_{i=1}^n$, $\boldsymbol{y}_{1:n} = (\boldsymbol{y}_i)_{i=1}^n$
SW order, number of slices : $p \in [1, +\infty)$, $n_S \in \mathbb{N}^*$
Bound parameter : $\lambda \in \mathbb{R}_+^*$
Number of iterations, learning rate : $T \in \mathbb{N}^*$, $\eta \in (0, 1)$
Initialized parameters : $(\mathbf{m}^{(0)}, \kappa^{(0)}) \in \mathbb{S}^{d-1} \times \mathbb{R}_+^*$

Output : Final parameters : $(\mathbf{m}^{(T)}, \kappa^{(T)})$

procedure PAC-Bayes-SW
  for $t \leftarrow 0$ to $T - 1$ **do**
    $\rho^{(t)} \leftarrow \text{vMF}(\mathbf{m}^{(t)}, \kappa^{(t)})$
    for $k \leftarrow 1$ to n_S **do**
      $\boldsymbol{\theta}_k^{(t)} \sim \rho^{(t)}$ [Davidson, 2018, Algorithm 1]
    end for
    $\rho_n^{(t)} \leftarrow n_S^{-1} \sum_{k=1}^n \delta_{\boldsymbol{\theta}_k^{(t)}}$
    $\mathcal{L}(x_{1:n}, y_{1:n}, \rho^{(t)}, \lambda) \leftarrow \text{SW}_p^p(\mu_n, \nu_n; \rho_n^{(t)}) - \lambda^{-1}\text{KL}(\rho^{(t)}||\rho^{(0)})$
    $\begin{bmatrix} \mathbf{m}^{(t+1)} \\ \kappa^{(t+1)} \end{bmatrix} \leftarrow \begin{bmatrix} \mathbf{m}^{(t)} \\ \kappa^{(t)} \end{bmatrix} + \eta \begin{bmatrix} \nabla_{\mathbf{m}}\mathcal{L}(x_{1:n}, y_{1:n}, \rho^{(t)}, \lambda) \\ \nabla_{\kappa}\mathcal{L}(x_{1:n}, y_{1:n}, \rho^{(t)}, \lambda) \end{bmatrix}$
  end for
  return $(\mathbf{m}^{(T)}, \kappa^{(T)})$
end procedure

7.8.2 Additional results

Figure 7.5 displays additional qualitative results for the generative modelling experiment described in Section 7.5.4. We observe that the images generated by DSW have a better quality than the ones produced by maxSW, even if DSW is not optimized at every training iteration.

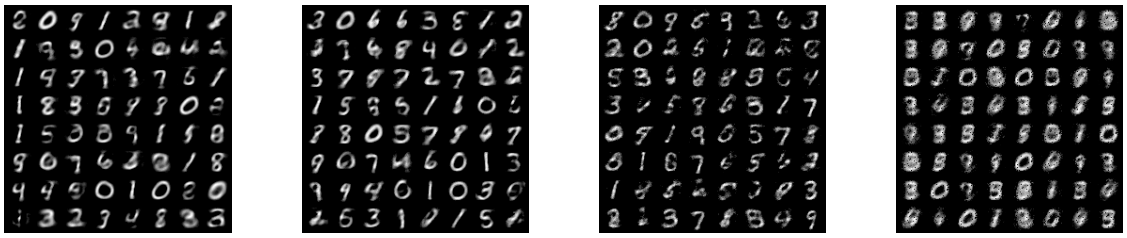


FIGURE 7.5 – Examples of generated MNIST digits. From left to right : DSW, DSW-10, maxSW, maxSW-10.

Conclusion and perspectives

In this thesis, we demonstrated diverse advantages of incorporating randomness in machine learning algorithms. In Chapter 3, we started by identifying reservoir computing with recurrent random features and computed their infinite limit, namely recurrent kernels. This allowed for more expressivity and a better scaling when a small number of points is available. We also scaled-up reservoir computing with respect to the number of neurons, by introducing structured transforms in the reservoir. We finished by proving the efficiency of our method on chaotic time-series prediction yielding state-of-the-art prediction capabilities at reduced computational and memory costs.

In Chapter 4, motivated by scaling up random transforms, we proposed to use an Optical Processing Unit (OPU) to compute the random matrix multiplication. This allowed the use of much larger input and output dimensions than GPUs. We computed the kernel limit of optical random features for different exponents of the feature map and compared speed-ups and savings in energy with a GPU.

We then introduced in Chapter 5 a new adversarial defense based on the parameter obfuscation provided by the OPU and trained it using Direct Feedback Alignment. We showed this yields adversarial robustness by design against white-box, black-box and transfer attacks when the network is trained from scratch. In a second part, we leveraged state-of-the-art robust models and improved their robustness by replacing their classifier with ROPUST. We benchmarked them on Robustbench and showed the increase in robustness comes at a very little cost in natural accuracy. We provided ablation studies and demonstrated that the whole defense block is required for maximum robustness.

In Chapter 6, by performing the random projection of Direct feedback Alignment using the OPU, we took into account the addition of experimental noise. We proved that this lead to a differentially private neural network. We computed the differential privacy parameters of the training algorithm and showed that, surprisingly, it held a good accuracy compared to backpropagation as the privacy (or noise) increased.

In Chapter 7, we used the PAC-Bayesian framework to optimize the distribution of random projections of Sliced-Wasserstein distances while being theoretically grounded. We computed the PAC-Bayes bounds for different cases of data distributions and numerically demonstrated their efficiency in maximizing the distance and in generative modelling.

To conclude this thesis, we showed that incorporating randomness at different parts of the

machine learning algorithm can lead to speed-ups at a small loss in accuracy. Indeed, supervised machine learning generally aims at finding good transformations of the data in a space where they are linearly separable. It is not essential to obtain the exact transformation for a perfect separation, but rather having an approximate separation at a smaller computational cost is often enough. This also holds for having approximate gradients given by Direct Feedback Alignment during the training of neural networks. On the other hand, as we have seen, adding randomness can yield security to systems, let it be adversarial robustness or differential privacy. Usually, adding this noise leads to a decrease of the accuracy of the neural network. However this decrease is reduced in our case, and we suspect this is due to a Direct Feedback Alignment training of the network. Indeed, as DFA leads to approximate backpropagation gradients, we can conjecture that adding some noise to these gradients may be less damageable.

An important feature of DFA that wasn't exploited in this thesis is the possible parallel update of the weights of the network. Indeed, we only used DFA for bypassing non-differentiable layers. In the case of very large models, this parallel update could lead to significant speed-ups in the training. This research direction could also inspire new training algorithms, where the backward pass is unlocked. One thing that is missing in the neural network literature is the unlocking of the forward pass, i.e. performing the transformation of all the layers in parallel. This could lead to even more efficient algorithms, and as a consequence the training and forward pass of machine learning models would be significantly faster.

Another interesting research direction is the study of the average transport plan of the Sliced-Wasserstein distance. Indeed, for each slice, we obtain a different matching/transport plan. Since these slices are a Monte-Carlo sampling of the uniform distribution on the sphere, we can hope that an infinite number of slices would yield an average transport plan. It would not be a permutation matrix, but rather a doubly-stochastic one, in the flavor of the Sinkhorn algorithm. Discovering a good heuristic for finding this average transport plan could yield another speed-up as we would not have to perform the Monte-Carlo sampling of the integral over the Sphere, but rather have an analytical formula.

Bibliography

[Abadi, 2016] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar et al. « Deep learning with differential privacy ». *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, p. 308-318 (cf. p. 91, 92, 98).

[Abay, 2018] Nazmiye Ceren Abay, Y. Zhou, Murat Kantarcioglu, B. Thuraisingham et L. Sweeney. « Privacy Preserving Synthetic Data Release Using Deep Learning ». *ECML/PKDD*. 2018 (cf. p. 92).

[Ács, 2019] G. Ács, Luca Melis, C. Castelluccia et Emiliano De Cristofaro. « Differentially Private Mixture of Generative Neural Networks ». *IEEE Transactions on Knowledge and Data Engineering* 31 (2019), p. 1109-1121 (cf. p. 92).

[Advani, 2020] Madhu S. Advani, Andrew M. Saxe et Haim Sompolinsky. « High-dimensional dynamics of generalization error in neural networks ». *Neural Networks* 132 (2020), p. 428-446 (cf. p. 19).

[Akrout, 2019a] Mohamed Akrout. *On the Adversarial Robustness of Neural Networks without Weight Transport*. 2019. arXiv : 1908.03560 [cs.LG] (cf. p. 70).

[Akrout, 2019b] Mohamed Akrout. « On the Adversarial Robustness of Neural Networks without Weight Transport ». *ArXiv* abs/1908.03560 (2019) (cf. p. 84).

[Alexandre Araujo, 2020] Rafael Pinot Alexandre Araujo Laurent Meunier et Benjamin Negrevergne. « Advocating for Multiple Defense Strategies against Adversarial Examples ». *Workshop on Machine Learning for CyberSecurity (MLCS@ECML-PKDD)* (2020) (cf. p. 73, 83).

[Alquier, 2021] Pierre Alquier. *User-friendly introduction to PAC-Bayes bounds*. 2021. arXiv : 2110.11216 [stat.ML] (cf. p. 27, 29, 109, 112).

[Ambroladze, 2007] Amiran Ambroladze, Emilio Parrado-hernández et John Shawe-taylor. « Tighter PAC-Bayes Bounds ». *Advances in Neural Information Processing Systems*. Sous la dir. de B. Schölkopf, J. Platt et T. Hoffman. T. 19. MIT Press, 2007 (cf. p. 109).

[Andriushchenko, 2019] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion et Matthias Hein. « Square attack : a query-efficient black-box adversarial attack via random search ». *arXiv preprint arXiv :1912.00049* (2019) (cf. p. 72, 73, 82, 83).

[Antonik, 2018] Piotr Antonik, Marvyn Gulina, Jaël Pauwels et Serge Massar. « Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronization and cryptography ». *Physical Review E* 98.1 (2018), p. 012215 (cf. p. 42).

[Antonik, 2015] Piotr Antonik, Anteo Smerieri, François Duport, Marc Haelterman et Serge Massar. « FPGA implementation of reservoir computing with online learning ». *24th Belgian-Dutch Conference on Machine Learning*. 2015 (cf. p. 34).

[Arjovsky, 2017] Martin Arjovsky, Soumith Chintala et Léon Bottou. « Wasserstein generative adversarial networks ». *International conference on machine learning*. PMLR. 2017, p. 214-223 (cf. p. 30, 108).

[Arjovsky, 2016] Martin Arjovsky, Amar Shah et Yoshua Bengio. « Unitary evolution recurrent neural networks ». *International Conference on Machine Learning*. 2016, p. 1120-1128 (cf. p. 37).

[Arpaia, 2021] Pasquale Arpaia, Gabriella Azzopardi, Frederic Blanc, Giuseppe Bregliozzi, Xavier Buffat, Loic Coyle et al. « Machine learning for beam dynamics studies at the CERN Large Hadron Collider ». *Nuclear Instruments and Methods in Physics Research Section A : Accelerators, Spectrometers, Detectors and Associated Equipment* 985 (2021), p. 164652 (cf. p. 2).

[Asoodeh, 2020] Shahab Asoodeh, Jiachun Liao, Flavio P Calmon, Oliver Kosut et Lalitha Sankar. « Three Variants of Differential Privacy : Lossless Conversion and Applications ». *arXiv preprint arXiv :2008.06529* (2020) (cf. p. 27, 96).

[Athalye, 2018a] Anish Athalye, Nicholas Carlini et David Wagner. « Obfuscated Gradients Give a False Sense of Security : Circumventing Defenses to Adversarial Examples ». *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*. 2018 (cf. p. 70, 72, 73).

[Athalye, 2018b] Anish Athalye, Nicholas Carlini et David Wagner. « Obfuscated Gradients Give a False Sense of Security : Circumventing Defenses to Adversarial Examples ». *Proceedings of the 35th International Conference on Machine Learning*. Sous la dir. de Jennifer Dy et Andreas Krause. T. 80. Proceedings of Machine Learning Research. Stockholmsmassan, Stockholm Sweden : PMLR, 2018, p. 274-283 (cf. p. 83, 85).

[Aubin, 2018] B. Aubin, A. Maillard, J. Barbier, F. Krzakala, N. Macris et L. Zdeborová. « The committee machine : Computational to statistical gaps in learning a two-layers neural network ». *Advances in Neural Information Processing Systems 31*. 2018, p. 3227-3238 (cf. p. 19).

[Bagdasaryan, 2019] Eugene Bagdasaryan, Omid Poursaeed et Vitaly Shmatikov. « Differential privacy has disparate impact on model accuracy ». *Advances in Neural Information Processing Systems* 32 (2019), p. 15479-15488 (cf. p. 103).

[Bahdanau, 2014] Dzmitry Bahdanau, Kyunghyun Cho et Yoshua Bengio. « Neural machine translation by jointly learning to align and translate ». *arXiv preprint arXiv :1409.0473* (2014) (cf. p. 17).

[Bahri, 2020] Y. Bahri, J. Kadmon, J. Pennington, S.S. Schoenholz, J. Sohl-Dickstein et S. Ganguli. « Statistical Mechanics of Deep Learning ». *Annual Review of Condensed Matter Physics* 11.1 (2020), p. 501-528 (cf. p. 19).

[Baity-Jesi, 2018] M. Baity-Jesi, L. Sagun, M. Geiger, S. Spigler, G.B. Arous, C. Cammarota et al. « Comparing Dynamics : Deep Neural Networks versus Glassy Systems ». *Proceedings of the 35th International Conference on Machine Learning*. 2018 (cf. p. 19).

[Balle, 2018] Borja Balle et Yu-Xiang Wang. « Improving the Gaussian Mechanism for Differential Privacy : Analytical Calibration and Optimal Denoising ». *Proceedings of the 35th International Conference on Machine Learning*. Sous la dir. de Jennifer Dy et Andreas Krause. T. 80. Proceedings of Machine Learning Research. Stockholmsmassan, Stockholm Sweden : PMLR, 2018, p. 394-403 (cf. p. 27, 96).

[Berger, 2019] Bonnie Berger et Hyunghoon Cho. *Emerging technologies towards enhancing privacy in genomic data sharing*. 2019 (cf. p. 91).

[Biehl, 1995] M. Biehl et H. Schwarze. « Learning by on-line gradient descent ». *J. Phys. A. Math. Gen.* 28.3 (1995), p. 643-656 (cf. p. 19, 20).

[Bobkov, 2019] Sergey G. Bobkov et Michel Ledoux. « One-dimensional empirical measures, order statistics, and Kantorovich transport distances ». *Memoirs of the American Mathematical Society* (2019) (cf. p. 114, 125).

[Bochner, 1933] Salomon Bochner. « Monotone funktionen, stieltjessche integrale und harmonische analyse ». *Mathematische Annalen* 108.1 (1933), p. 378-410 (cf. p. 11).

[Bonet, 2021] Clément Bonet, Nicolas Courty, François Septier et Lucas Drumetz. « Sliced-Wasserstein Gradient Flows ». *arXiv preprint arXiv :2110.10972* (2021) (cf. p. 109).

[Bonneel, 2015] Nicolas Bonneel, Julien Rabin, Gabriel Peyré et Hanspeter Pfister. « Sliced and radon wasserstein barycenters of measures ». *Journal of Mathematical Imaging and Vision* 51.1 (2015), p. 22-45 (cf. p. 32, 109, 111).

[Boucheron, 2003] Stéphane Boucheron, Gábor Lugosi et Olivier Bousquet. « Concentration inequalities ». *Summer School on Machine Learning*. Springer. 2003, p. 208-240 (cf. p. 38).

[Bousquet, 2017] Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl-Johann Simon-Gabriel et Bernhard Schoelkopf. « From optimal transport to generative modeling : the VEGAN cookbook ». *arXiv preprint arXiv :1705.07642* (2017) (cf. p. 108).

[Brossollet, 2021] Charles Brossollet, Alessandro Cappelli, Igor Carron, Charidimos Chaintoutis, Amélie Chatelain, Laurent Daudet et al. « LightOn Optical Processing Unit : Scaling-up AI and HPC with a Non von Neumann co-processor ». *arXiv preprint arXiv :2107.11814* (2021) (cf. p. 7, 22).

[Brown, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal et al. « Language models are few-shot learners ». *Advances in neural information processing systems* 33 (2020), p. 1877-1901 (cf. p. 2).

[Brutzkus, 2017] Alon Brutzkus et Amir Globerson. « Globally Optimal Gradient Descent for a ConvNet with Gaussian Inputs ». *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. 2017, p. 605-614 (cf. p. 19).

[Bubeck, 2019] Sébastien Bubeck, Yin Tat Lee, Eric Price et Ilya Razenshteyn. « Adversarial examples from computational constraints ». *International Conference on Machine Learning*. PMLR. 2019, p. 831-840 (cf. p. 69).

[Campbell, 2002] Murray Campbell, A Joseph Hoane Jr et Feng-hsiung Hsu. « Deep blue ». *Artificial intelligence* 134.1-2 (2002), p. 57-83 (cf. p. 1).

[Caponnetto, 2007] Andrea Caponnetto et Ernesto De Vito. « Optimal rates for the regularized least-squares algorithm ». *Foundations of Computational Mathematics* 7.3 (2007), p. 331-368 (cf. p. 53).

[Cappelli, 2021a] A. Cappelli, Ruben Ohana, Julien Launay, Laurent Meunier, Iacopo Poli et F. Krzakala. « Adversarial Robustness by Design through Analog Computing and Synthetic Gradients ». *ArXiv* abs/2101.02115 (2021) (cf. p. 4, 6, 69).

[Cappelli, 2021b] Alessandro Cappelli, Julien Launay, Laurent Meunier, Ruben Ohana et Iacopo Poli. « Ropust : Improving robustness through fine-tuning with photonic processors and synthetic gradients ». *arXiv preprint arXiv :2108.04217* (2021) (cf. p. 4, 6, 69).

[Carlini, 2017] Nicholas Carlini et David Wagner. « Towards evaluating the robustness of neural networks ». *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, p. 39-57 (cf. p. 72, 82).

[Carmon, 2019] Y. Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang et John C. Duchi. « Unlabeled Data Improves Adversarial Robustness ». *NeurIPS*. 2019 (cf. p. 85, 86).

[Carratino, 2018] Luigi Carratino, Alessandro Rudi et Lorenzo Rosasco. « Learning with SGD and random features ». *Advances in Neural Information Processing Systems*. 2018, p. 10192-10203 (cf. p. 34).

[Carriere, 2017] Mathieu Carriere, Marco Cuturi et Steve Oudot. « Sliced Wasserstein kernel for persistence diagrams ». *International conference on machine learning*. PMLR. 2017, p. 664-673 (cf. p. 109).

[Catoni, 2007] O. Catoni. *Pac-Bayesian Supervised Classification : The Thermodynamics of Statistical Learning*. T. 56. Institute of Mathematical Statistics, 2007 (cf. p. 109, 112).

[Catoni, 2003] Olivier Catoni. *A PAC-Bayesian approach to adaptive classification*. preprint LPMA 840. 2003 (cf. p. 30, 112, 121).

[Chang, 2020] Hanten Chang et Katsuya Futagami. « Reinforcement learning with convolutional reservoir computing ». *Applied Intelligence* (2020), p. 1-11 (cf. p. 34).

[Chanyaswad, 2018] Thee Chanyaswad, Alex Dytso, H Vincent Poor et Prateek Mittal. « Mvg mechanism : Differential privacy under matrix-valued query ». *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, p. 230-246 (cf. p. 99).

[Chen, 2019] Dexiong Chen, Laurent Jacob et Julien Mairal. « Recurrent Kernel Networks ». *Advances in Neural Information Processing Systems*. 2019, p. 13431-13442 (cf. p. 34).

[Chen, 2017] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi et Cho-Jui Hsieh. « Zoo : Zeroth order optimization based black-box attacks to deep neural networks without training substitute models ». *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM. 2017, p. 15-26 (cf. p. 73, 82).

[Cherapanamjeri, 2022] Yeshwanth Cherapanamjeri et Jelani Nelson. « Uniform Approximations for Randomized Hadamard Transforms with Applications ». *arXiv preprint arXiv :2203.01599* (2022) (cf. p. 14).

[Chizat, 2018] L. Chizat et F. Bach. « On the Global Convergence of Gradient Descent for Over-parameterized Models using Optimal Transport ». *Advances in Neural Information Processing Systems 31*. 2018, p. 3040-3050 (cf. p. 19).

[Choromanski, 2017a] Krzysztof Choromanski, Mark Rowland et Adrian Weller. « The Unreasonable Effectiveness of Structured Random Orthogonal Embeddings ». *Advances in Neural Information Processing Systems 31*. Long Beach, California, USA : Curran Associates Inc., 2017, p. 218-227 (cf. p. 14).

[Choromanski, 2017b] Krzysztof M Choromanski, Mark Rowland et Adrian Weller. « The unreasonable effectiveness of structured random orthogonal embeddings ». *Advances in Neural Information Processing Systems*. 2017, p. 219-228 (cf. p. 40).

[Cohen, 2019] Jeremy M. Cohen, Elan Rosenfeld et J. Zico Kolter. « Certified Adversarial Robustness via Randomized Smoothing ». *arXiv preprint arXiv :1902.02918* abs/1902.02918 (2019) (cf. p. 73, 83).

[Courty, 2016] Nicolas Courty, Rémi Flamary, Devis Tuia et Alain Rakotomamonjy. « Optimal transport for domain adaptation ». *IEEE transactions on pattern analysis and machine intelligence* 39.9 (2016), p. 1853-1865 (cf. p. 108).

[Croce, 2020a] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Nicolas Flammarion, Mung Chiang, Prateek Mittal et al. « RobustBench : a standardized adversarial robustness benchmark ». *arXiv preprint arXiv :2010.09670* (2020) (cf. p. 25, 70).

[Croce, 2020b] Francesco Croce et Matthias Hein. « Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack ». *ICML*. 2020 (cf. p. 83).

[Croce, 2020c] Francesco Croce et Matthias Hein. « Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks ». *International Conference on Machine Learning*. 2020 (cf. p. 25).

[Croce, 2020d] Francesco Croce et Matthias Hein. « Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks ». *ICML*. 2020 (cf. p. 70, 82, 83).

[Cutajar, 2017] Kurt Cutajar, Edwin Bonilla, Pietro Michiardi et Maurizio Filippone. « Random feature expansions for deep Gaussian processes ». *ICML 2017, 34th International Conference on Machine Learning, 6-11 August 2017, Sydney, Australia*. S, 2017 (cf. p. 53).

[Cuturi, 2013] Marco Cuturi. « Sinkhorn distances : Lightspeed computation of optimal transport ». *Advances in neural information processing systems* 26 (2013), p. 2292-2300 (cf. p. 31, 109).

[Cuturi, 2016] Marco Cuturi et Gabriel Peyré. « A smoothed dual approach for variational Wasserstein problems ». *SIAM Journal on Imaging Sciences* 9.1 (2016), p. 320-343 (cf. p. 109).

[Davidson, 2018] Tim R. Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf et Jakub M. Tomczak. « Hyperspherical Variational Auto-Encoders ». *34th Conference on Uncertainty in Artificial Intelligence (UAI-18)* (2018) (cf. p. 117, 130).

[Degrave, 2022] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese et al. « Magnetic control of tokamak plasmas through deep reinforcement learning ». *Nature* 602.7897 (2022), p. 414-419 (cf. p. 2).

[Deshpande, 2019] Ishan Deshpande, Yuan-Ting Hu, Ruoyu Sun, Ayis Pyrros, Nasir Siddiqui, Sanmi Koyejo et al. « Max-sliced wasserstein distance and its use for gans ». *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, p. 10648-10656 (cf. p. 109, 111).

[Deshpande, 2018] Ishan Deshpande, Ziyu Zhang et Alexander G Schwing. « Generative modeling using the sliced wasserstein distance ». *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, p. 3483-3491 (cf. p. 109, 119).

[Devlin, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee et Kristina Toutanova. « Bert : Pre-training of deep bidirectional transformers for language understanding ». *arXiv preprint arXiv :1810.04805* (2018) (cf. p. 2).

[Ding, 2020] Gavin Weiguang Ding, Yash Sharma, Kry Yik-Chau Lui et Ruitong Huang. « Max-Margin Adversarial (MMA) Training : Direct Input Space Margin Maximization through Adversarial Training ». *ArXiv* abs/1812.02637 (2020) (cf. p. 85, 86).

[Dong, 2018] Jonathan Dong, Sylvain Gigan, Florent Krzakala et Gilles Wainrib. « Scaling up echo-state networks with multiple light scattering ». *2018 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE. 2018, p. 448-452 (cf. p. 34, 54).

[Dong, 2020] Jonathan Dong, Ruben Ohana, Mushegh Rafayelyan et Florent Krzakala. « Reservoir computing meets recurrent kernels and structured transforms ». *Advances in Neural Information Processing Systems* 33 (2020), p. 16785-16796 (cf. p. 4, 6, 33).

[Dong, 2019] Jonathan Dong, Mushegh Rafayelyan, Florent Krzakala et Sylvain Gigan. « Optical Reservoir Computing using multiple light scattering for chaotic systems prediction ». *IEEE Journal of Selected Topics in Quantum Electronics* 26.1 (2019), p. 1-12 (cf. p. 34, 54).

[Donsker, 1975] Monroe D Donsker et SR Srinivasa Varadhan. « Asymptotic evaluation of certain Markov process expectations for large time, I ». *Communications on Pure and Applied Mathematics* 28.1 (1975), p. 1-47 (cf. p. 121).

[Du, 2018] Simon Du, Jason Lee, Yuandong Tian, Aarti Singh et Barnabas Poczos. « Gradient Descent Learns One-hidden-layer CNN : Don't be Afraid of Spurious Local Minima ». *Proceedings of the 35th International Conference on Machine Learning*. T. 80. 2018, p. 1339-1348 (cf. p. 19).

[Duport, 2012] François Duport, Bendix Schneider, Anteo Smerieri, Marc Haelterman et Serge Massar. « All-optical reservoir computing ». *Optics express* 20.20 (2012), p. 22783-22795 (cf. p. 34).

[Dwork, 2006a] C. Dwork, F. McSherry, Kobbi Nissim et A. D. Smith. « Calibrating Noise to Sensitivity in Private Data Analysis ». *TCC*. 2006 (cf. p. 92, 95).

[Dwork, 2008] Cynthia Dwork. « Differential privacy : A survey of results ». *International conference on theory and applications of models of computation*. Springer. 2008, p. 1-19 (cf. p. 26, 95).

[Dwork, 2006b] Cynthia Dwork, Frank McSherry, Kobbi Nissim et Adam Smith. « Calibrating Noise to Sensitivity in Private Data Analysis ». *Theory of Cryptography*. Springer Berlin Heidelberg, 2006, p. 265-284 (cf. p. 26).

[Embrechts, 2013] Paul Embrechts, Claudia Klüppelberg et Thomas Mikosch. *Modelling extremal events : for insurance and finance*. T. 33. Springer Science & Business Media, 2013 (cf. p. 115).

[Engel, 2001] Andreas Engel et Christian Van den Broeck. *Statistical mechanics of learning*. Cambridge University Press, 2001 (cf. p. 19, 20).

[Eykholt, 2018] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao et al. « Robust physical-world attacks on deep learning visual classification ». *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, p. 1625-1634 (cf. p. 3).

[Fino, 1976] B. J. Fino et V. R. Algazi. « Unified Matrix Treatment of the Fast Walsh-Hadamard Transform ». *IEEE Transactions on Computers* 25.11 (1976), p. 1142-1146 (cf. p. 13).

[Frenkel, 2019] Charlotte Frenkel, Martin Lefebvre et David Bol. « Learning without feedback : Direct random target projection as a feedback-alignment algorithm with layerwise feedforward training ». 2019. arXiv : 1909.01311 (cf. p. 19).

[Frenkel, 2021] Charlotte Frenkel, Martin Lefebvre et David Bol. « Learning without feedback : Fixed random learning signals allow for feedforward training of deep neural networks ». *Frontiers in neuroscience* 15 (2021) (cf. p. 101).

[Frogner, 2015] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya-Polo et Tomaso Poggio. « Learning with a Wasserstein loss ». *arXiv preprint arXiv :1506.05439* (2015) (cf. p. 108).

[Gabrié, 2020] Marylou Gabrié. « Mean-field inference methods for neural networks ». *Journal of Physics A : Mathematical and Theoretical* 53.22 (2020), p. 223002 (cf. p. 19).

[Gallicchio, 2020] Claudio Gallicchio et Simone Scardapane. « Deep Randomized Neural Networks ». *Recent Trends in Learning From Data*. Springer, 2020, p. 43-68 (cf. p. 34).

[Garcelon, 2020] Evrard Garcelon, Baptiste Roziere, Laurent Meunier, Jean Tarbouriech, Olivier Teytaud, Alessandro Lazaric et al. « Adversarial Attacks on Linear Contextual Bandits ». *Advances in Neural Information Processing Systems* 33 (2020) (cf. p. 69).

[Gardner, 1989] E. Gardner et B. Derrida. « Three unfinished works on the optimal storage capacity of networks ». *Journal of Physics A : Mathematical and General* 22.12 (1989), p. 1983-1994 (cf. p. 19).

[George, 2018] Daniel George et Eliu Antonio Huerta. « Deep Learning for real-time gravitational wave detection and parameter estimation : Results with Advanced LIGO data ». *Physics Letters B* 778 (2018), p. 64-70 (cf. p. 2).

[Germain, 2009] P. Germain, A. Lacasse, F. Laviolette et M. Marchand. « PAC-Bayesian Learning of Linear Classifiers ». *Proc. of the 26th International Conference on Machine Learning (ICML)*. 2009, p. 353-360 (cf. p. 109).

[Ghimire, 2021] Sandesh Ghimire, Aria Masoomi et Jennifer Dy. « Reliable Estimation of KL Divergence using a Discriminator in Reproducing Kernel Hilbert Space ». *Advances in Neural Information Processing Systems* 34 (2021) (cf. p. 116).

[Ghorbani, 2019] B. Ghorbani, S. Mei, T. Misiakiewicz et A. Montanari. « Limitations of Lazy Training of Two-layers Neural Network ». *Advances in Neural Information Processing Systems 32*. 2019, p. 9111-9121 (cf. p. 19).

[Goibert, 2019] Morgane Goibert et Elvis Dohmatob. « Adversarial Robustness via Adversarial Label-Smoothing ». *ArXiv* abs/1906.11567 (2019) (cf. p. 90).

[Goldt, 2019] S. Goldt, M.S. Advani, A.M. Saxe, F. Krzakala et L. Zdeborová. « Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup ». *Advances in Neural Information Processing Systems 32*. 2019 (cf. p. 19).

[Goodfellow, 2015] I. Goodfellow, Jonathon Shlens et Christian Szegedy. « Explaining and Harnessing Adversarial Examples ». *CoRR* abs/1412.6572 (2015) (cf. p. 24, 69, 72, 73, 75, 82).

[Gowal, 2020] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy A. Mann et P. Kohli. « Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples ». *ArXiv* abs/2010.03593 (2020) (cf. p. 85, 86).

[Guo, 2019] Xianxin Guo, TD Barrett, ZM Wang et AI Lvovsky. « End-to-end optical backpropagation for training neural networks » (2019) (cf. p. 102).

[Gupta, 2020a] S. Gupta, R. Gribonval, L. Daudet et Ivan Dokmani'c. « Fast Optical System Identification by Numerical Interferometry ». *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), p. 1474-1478 (cf. p. 87, 88).

[Gupta, 2019a] S. Gupta, R. Gribonval, L. Daudet et Ivan Dokmanić. « Don't take it lightly : Phasing optical random projections with unknown operators ». *NeurIPS*. 2019 (cf. p. 87).

[Gupta, 2019b] Sidharth Gupta, Rémi Gribonval, Laurent Daudet et Ivan Dokmanić. « Don't take it lightly : Phasing optical random projections with unknown operators ». *Advances in Neural Information Processing Systems*. 2019, p. 14855-14865 (cf. p. 70, 74).

[Gupta, 2020b] Sidharth Gupta, Rémi Gribonval, Laurent Daudet et Ivan Dokmanić. « Fast Optical System Identification by Numerical Interferometry ». *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, p. 1474-1478 (cf. p. 70, 74).

[Haddouche, 2021] Maxime Haddouche, Benjamin Guedj, Omar Rivasplata et John Shawe-Taylor. « PAC-Bayes unleashed : generalisation bounds with unbounded losses ». *Entropy* 23.10 (2021), p. 1330 (cf. p. 28, 113).

[Hasnat, 2017] Md Hasnat, Julien Bohné, Jonathan Milgram, Stéphane Gentric, Liming Chen et al. « von mises-fisher mixture model-based deep learning : Application to face verification ». *arXiv preprint arXiv :1706.04264* (2017) (cf. p. 117).

[Hawes, 2020] Michael B Hawes. « Implementing differential privacy : seven lessons from the 2020 United States Census » (2020) (cf. p. 3).

[He, 2016] Kaiming He, X. Zhang, Shaoqing Ren et Jian Sun. « Identity Mappings in Deep Residual Networks ». *ArXiv* abs/1603.05027 (2016) (cf. p. 86).

[He, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. « Deep Residual Learning for Image Recognition ». *arXiv preprint arXiv :1512.03385* (2015) (cf. p. 58).

[Hendrycks, 2019] Dan Hendrycks, Kimin Lee et Mantas Mazeika. « Using Pre-Training Can Improve Model Robustness and Uncertainty ». *ICML*. 2019 (cf. p. 85, 86).

[Hermans, 2012] Michiel Hermans et Benjamin Schrauwen. « Recurrent kernel machines : Computing with infinite echo state networks ». *Neural Computation* 24.1 (2012), p. 104-133 (cf. p. 34).

[Hesslow, 2021] Daniel Hesslow, Alessandro Cappelli, Igor Carron, Laurent Daudet, Raphaël Lafargue, Kilian Müller et al. « Photonic co-processors in HPC : using LightOn OPUs for Randomized Numerical Linear Algebra ». *arXiv preprint arXiv :2104.14429* (2021) (cf. p. 7).

[Hinaut, 2013] Xavier Hinaut et Peter Ford Dominey. « Real-time parallel processing of grammatical structure in the fronto-striatal system : A recurrent network simulation study using reservoir computing ». *PloS one* 8.2 (2013) (cf. p. 34).

[Hooker, 2020] Sara Hooker. « The Hardware Lottery ». *ArXiv* abs/2009.06489 (2020) (cf. p. 90).

[Hooker, 2021] Sara Hooker. « Moving beyond "algorithmic bias is a data problem" ». *Patterns* 2.4 (2021), p. 100241 (cf. p. 103).

[Hughes, 2018] Tyler W Hughes, Momchil Minkov, Yu Shi et Shanhui Fan. « Training of photonic neural networks through in situ backpropagation and gradient measurement ». *Optica* 5.7 (2018), p. 864-871 (cf. p. 102).

[Ilyas, 2018a] Andrew Ilyas, Logan Engstrom, Anish Athalye et Jessy Lin. « Black-box adversarial attacks with limited queries and information ». *arXiv preprint arXiv :1804.08598* (2018) (cf. p. 24, 73, 82).

[Ilyas, 2018b] Andrew Ilyas, Logan Engstrom et Aleksander Madry. « Prior convictions : Black-box adversarial attacks with bandits and priors ». *arXiv preprint arXiv :1807.07978* (2018) (cf. p. 24, 73, 77, 82).

[Inubushi, 2017] Masanobu Inubushi et Kazuyuki Yoshimura. « Reservoir computing beyond memory-nonlinearity trade-off ». *Scientific reports* 7.1 (2017), p. 1-10 (cf. p. 41).

[Jacot, 2018] A. Jacot, F. Gabriel et C. Hongler. « Neural tangent kernel : Convergence and generalization in neural networks ». *Advances in Neural Information Processing Systems 32*. 2018, p. 8571-8580 (cf. p. 34).

[Jaeger, 2001] Herbert Jaeger. « The "echo state" approach to analysing and training recurrent neural networks-with an erratum note ». *Bonn, Germany : German National Research Center for Information Technology GMD Technical Report* 148.34 (2001), p. 13 (cf. p. 33, 41).

[Jeddi, 2020] Ahmadreza Jeddi, M. Shafiee et A. Wong. « A Simple Fine-tuning Is All You Need : Towards Robust Deep Learning Via Adversarial Fine-tuning ». *ArXiv* abs/2012.13628 (2020) (cf. p. 83).

[Jin, 2017] Yingyezhe Jin et Peng Li. « Performance and robustness of bio-inspired digital liquid state machines : A case study of speech recognition ». *Neurocomputing* 226 (2017), p. 145-160 (cf. p. 34).

[Johnson, 2018] Noah Johnson, Joseph P Near et Dawn Song. « Towards practical differential privacy for SQL queries ». *Proceedings of the VLDB Endowment* 11.5 (2018), p. 526-539 (cf. p. 91).

[Jumper, 2021] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger et al. « Highly accurate protein structure prediction with AlphaFold ». *Nature* 596.7873 (2021), p. 583-589 (cf. p. 2).

[Kantorovitch, 1958] Leonid Kantorovitch. « On the translocation of masses ». *Management science* 5.1 (1958), p. 1-4 (cf. p. 30).

[Kar, 2012] Purushottam Kar et Harish Karnick. « Random feature maps for dot product kernels ». *Artificial Intelligence and Statistics*. 2012, p. 583-591 (cf. p. 36).

[Keriven, 2018] Nicolas Keriven, Damien Garreau et Iacopo Poli. « NEWMA : a new method for scalable model-free online change-point detection ». *arXiv preprint arXiv :1805.08061* (2018) (cf. p. 54).

[Kingma, 2014] Diederik P Kingma et Jimmy Ba. « Adam : A method for stochastic optimization ». *arXiv preprint arXiv :1412.6980* (2014) (cf. p. 85).

[Kingma, 2015] Diederik P. Kingma et Jimmy Ba. « Adam : A Method for Stochastic Optimization ». *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Sous la dir. d'Yoshua Bengio et Yann LeCun. 2015 (cf. p. 118).

[Kinzel, 1990] W. Kinzel et P. Ruján. « Improving a Network Generalization Ability by Selecting Examples ». *EPL (Europhysics Letters)* 13.5 (1990), p. 473-477 (cf. p. 19).

[Kolouri, 2019a] S. Kolouri, K. Nadjahi, U. Simsekli, R. Badeau et G. Rohde. « Generalized Sliced Wasserstein Distances ». *Advances in Neural Information Processing Systems 32*. 2019, p. 261-272 (cf. p. 109).

[Kolouri, 2020] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli et Shahin Shahrampour. « Generalized sliced distances for probability distributions ». *arXiv preprint arXiv :2002.12537* (2020) (cf. p. 109).

[Kolouri, 2017] Soheil Kolouri, Se Rim Park, Matthew Thorpe, Dejan Slepcev et Gustavo K Rohde. « Optimal mass transport : Signal processing and machine-learning applications ». *IEEE signal processing magazine* 34.4 (2017), p. 43-59 (cf. p. 108).

[Kolouri, 2019b] Soheil Kolouri, Phillip E Pope, Charles E Martin et Gustavo K Rohde. « Sliced Wasserstein auto-encoders ». *International Conference on Learning Representations*. 2019 (cf. p. 109).

[Kolouri, 2018] Soheil Kolouri, Gustavo K Rohde et Heiko Hoffmann. « Sliced wasserstein distance for learning gaussian mixture models ». *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, p. 3427-3436 (cf. p. 109).

[Kolouri, 2016] Soheil Kolouri, Yang Zou et Gustavo K Rohde. « Sliced Wasserstein kernels for probability distributions ». *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, p. 5258-5267 (cf. p. 109).

[Konnik, 2014] Mikhail Konnik et James Welsh. « High-level numerical simulations of noise in CCD and CMOS photosensors : review and tutorial ». *arXiv preprint arXiv :1412.4031* (2014) (cf. p. 97).

[Kontorovich, 2014] Aryeh Kontorovich. « Concentration in unbounded metric spaces and algorithmic stability ». *Proceedings of the 31st International Conference on Machine Learning*. Sous la dir. d'Eric P. Xing et Tony Jebara. T. 32. Proceedings of Machine Learning Research 2. Bejing, China : PMLR, 2014, p. 28-36 (cf. p. 114, 115, 126, 127).

[Kozyrskiy, 2021] B Kozyrskiy, I Poli, R Ohana, L Daudet, I Carron et M Filippone. « Binarization for Optical Processing Units via REINFORCE » (2021) (cf. p. 23).

[Krizhevsky, 2009a] A. Krizhevsky. « Learning Multiple Layers of Features from Tiny Images ». *Learning Multiple Layers of Features from Tiny Images*. 2009 (cf. p. 85).

[Krizhevsky, 2009b] Alex Krizhevsky, Vinod Nair et Geoffrey Hinton. « Cifar-10 and cifar-100 datasets ». *URl : https ://www. cs. toronto. edu/kriz/cifar. html* 6 (2009) (cf. p. 72).

[Krizhevsky, 2012] Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton. « ImageNet Classification with Deep Convolutional Neural Networks ». *Advances in Neural Information Processing Systems 25*. Sous la dir. de F. Pereira, C. J. C. Burges, L. Bottou et K. Q. Weinberger. Curran Associates, Inc., 2012, p. 1097-1105 (cf. p. 58).

[Kumar, 2020] K Naveen Kumar, C Vishnu, Reshmi Mitra et C Krishna Mohan. « Black-box adversarial attacks in autonomous vehicle technology ». *2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*. IEEE. 2020, p. 1-7 (cf. p. 24).

[Kumar, 2018] Sachin Kumar et Yulia Tsvetkov. « Von mises-fisher loss for training sequence to sequence models with continuous outputs ». *arXiv preprint arXiv :1812.04616* (2018) (cf. p. 117).

[Kurakin, 2016] Alexey Kurakin, Ian Goodfellow et Samy Bengio. « Adversarial examples in the physical world ». *arXiv preprint arXiv :1607.02533* (2016) (cf. p. 72, 82).

[Kuramoto, 1978] Yoshiki Kuramoto. « Diffusion-induced chaos in reaction systems ». *Progress of Theoretical Physics Supplement* 64 (1978), p. 346-367 (cf. p. 42).

[Larger, 2012] Laurent Larger, Miguel C Soriano, Daniel Brunner, Lennert Appeltant, Jose M Gutiérrez, Luis Pesquera et al. « Photonic information processing beyond Turing : an optoelectronic implementation of reservoir computing ». *Optics express* 20.3 (2012), p. 3241-3249 (cf. p. 34).

[Launay, 2020a] Julien Launay, Iacopo Poli, François Boniface et Florent Krzakala. « Direct Feedback Alignment Scales to Modern Deep Learning Tasks and Architectures ». *Advances in Neural Information Processing Systems* 33 (2020) (cf. p. 17, 18, 74, 75, 93).

[Launay, 2020b] Julien Launay, Iacopo Poli, Kilian Müller, Gustave Pariente, Igor Carron, Laurent Daudet et al. « Hardware Beyond Backpropagation : a Photonic Co-Processor for Direct Feedback Alignment ». *Beyond Backpropagation Workshop, NeurIPS 2020* (2020) (cf. p. 101).

[Laviolette, 2006] François Laviolette, Mario Marchand et Mohak Shah. « A PAC-Bayes approach to the Set Covering Machine ». *Advances in Neural Information Processing Systems*. Sous la dir. d'Y. Weiss, B. Schölkopf et J. Platt. T. 18. MIT Press, 2006 (cf. p. 109).

[Le, 2013] Quoc Le, Tamás Sarlós et Alexander Smola. « Fastfood-computing Hilbert space expansions in loglinear time ». *International Conference on Machine Learning*. 2013, p. 244-252 (cf. p. 13, 34, 37, 54).

[LeCun, 2015] Y. LeCun, Y. Bengio et G.E. Hinton. « Deep learning ». *Nature* 521.7553 (2015), p. 436-444 (cf. p. 1).

[LeCun, 1990] Y. LeCun, B.E. Boser, J.S. Denker, D. Henderson, R.E. Howard, W.E. Hubbard et al. « Handwritten digit recognition with a back-propagation network ». *Advances in neural information processing systems*. 1990, p. 396-404 (cf. p. 11).

[Lecun, 1998] Y. Lecun, L. Bottou, Y. Bengio et P. Haffner. « Gradient-based learning applied to document recognition ». *Proceedings of the IEEE* 86.11 (1998), p. 2278-2324 (cf. p. 17).

[Lecuyer, 2018] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu et S. Jana. « Certified Robustness to Adversarial Examples with Differential Privacy ». *2019 IEEE Symposium on Security and Privacy (SP)*. 2018, p. 727-743 (cf. p. 73, 83).

[Lee, 2020] Jaewoo Lee et Daniel Kifer. « Differentially Private Deep Learning with Direct Feedback Alignment ». *CoRR* abs/2010.03701 (2020). arXiv : 2010.03701 (cf. p. 92, 93, 99, 100).

[Lei, 2020] Jing Lei. « Convergence and concentration of empirical measures under Wasserstein distance in unbounded functional spaces ». *Bernoulli* 26.1 (2020), p. 767-798 (cf. p. 115, 127).

[Liang, 2019] Kevin Liang, Guoyin Wang, Yitong Li, Ricardo Henao et Lawrence Carin. « Kernel-Based Approaches for Sequence Modeling : Connections to Neural Methods ». *Advances in Neural Information Processing Systems*. 2019, p. 3387-3398 (cf. p. 34).

[Liao, 2018] Zhenyu Liao et Romain Couillet. « On the spectrum of random features maps of high dimensional data ». *arXiv preprint arXiv :1805.11916* (2018) (cf. p. 12, 36).

[LightOn, 2020] LightOn. *Photonic Computing for Massively Parallel AI - A White Paper, v1.0*. https://lighton.ai/wp-content/uploads/2020/05/LightOn-White-Paper-v1.0.pdf. 2020 (cf. p. 92, 95).

[Lillicrap, 2014] T. Lillicrap, D. Cownden, D. Tweed et C. Akerman. « Random feedback weights support learning in deep neural networks ». *ArXiv* abs/1411.0247 (2014) (cf. p. 17, 84).

[Lillicrap, 2016] T.P. Lillicrap, D. Cownden, D.B. Tweed et C.J. Akerman. « Random synaptic feedback weights support error backpropagation for deep learning ». *Nature Communications* 7 (2016), p. 1-10 (cf. p. 17, 19, 93).

[Lin, 2021] Tianyi Lin, Zeyu Zheng, Elynn Y. Chen, Marco Cuturi et Michael I. Jordan. « On Projection Robust Optimal Transport : Sample Complexity and Model Misspecification ». *AISTATS*. 2021, p. 262-270 (cf. p. 111, 124).

[Liu, 2020] Fanghui Liu, Xiaolin Huang, Yudong Chen et Johan AK Suykens. « Random Features for Kernel Approximation : A Survey in Algorithms, Theory, and Beyond ». *arXiv preprint arXiv :2004.11154* (2020) (cf. p. 34).

[Liutkus, 2014] Antoine Liutkus, David Martina, Sébastien Popoff, Gilles Chardon, Ori Katz, Geoffroy Lerosey et al. « Imaging with nature : Compressive imaging using a multiply scattering medium ». *Scientific reports* 4 (2014), p. 5552 (cf. p. 22, 55).

[Liutkus, 2019] Antoine Liutkus, Umut Simsekli, Szymon Majewski, Alain Durmus et Fabian-Robert Stöter. « Sliced-Wasserstein flows : Nonparametric generative modeling via optimal transport and diffusions ». *International Conference on Machine Learning*. PMLR. 2019, p. 4104-4113 (cf. p. 109).

[Lukoševičius, 2009] Mantas Lukoševičius et Herbert Jaeger. « Reservoir computing approaches to recurrent neural network training ». *Computer Science Review* 3.3 (2009), p. 127-149 (cf. p. 34).

[Lukoševičius, 2012] Mantas Lukoševičius, Herbert Jaeger et Benjamin Schrauwen. « Reservoir computing trends ». *KI-Künstliche Intelligenz* 26.4 (2012), p. 365-371 (cf. p. 34).

[Madry, 2018a] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras et Adrian Vladu. « Towards Deep Learning Models Resistant to Adversarial Attacks ». *International Conference on Learning Representations*. 2018 (cf. p. 24, 25, 75).

[Madry, 2018b] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras et Adrian Vladu. « Towards Deep Learning Models Resistant to Adversarial Attacks ». *International Conference on Learning Representations*. 2018 (cf. p. 70, 72, 73, 82).

[Manole, 2020] Tudor Manole, Sivaraman Balakrishnan et Larry Wasserman. *Minimax Confidence Intervals for the Sliced Wasserstein Distance*. 2020. arXiv : 1909.07862 [math.ST] (cf. p. 114, 115, 124, 128).

[McAllester, 1999] David A McAllester. « Some pac-bayesian theorems ». *Machine Learning* 37.3 (1999), p. 355-363 (cf. p. 109, 112).

[McDiarmid, 1989] Colin McDiarmid. « On the method of bounded differences ». *Surveys in combinatorics* 141.1 (1989), p. 148-188 (cf. p. 113, 114, 122).

[Mei, 2019] S. Mei et A. Montanari. « The generalization error of random features regression : Precise asymptotics and double descent curve ». 2019. arXiv : 1908.05355 (cf. p. 34).

[Mei, 2018] S. Mei, A. Montanari et P. Nguyen. « A mean field view of the landscape of two-layer neural networks ». *Proceedings of the National Academy of Sciences* 115.33 (2018), E7665-E7671 (cf. p. 19).

[Mena, 2019] Gonzalo Mena et Jonathan Niles-Weed. « Statistical bounds for entropic optimal transport : sample complexity and the central limit theorem ». *Advances in Neural Information Processing Systems*. Sous la dir. de H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox et R. Garnett. T. 32. Curran Associates, Inc., 2019 (cf. p. 114, 127).

[Meunier, 2019] Laurent Meunier, Jamal Atif et Olivier Teytaud. « Yet another but more efficient black-box adversarial attack : tiling and evolution strategies ». *arXiv preprint arXiv :1910.02244* (2019) (cf. p. 73, 82).

[Mironov, 2017] I. Mironov. « Rényi Differential Privacy ». *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. 2017, p. 263-275 (cf. p. 26, 27, 93, 95, 96).

[Mironov, 2019] Ilya Mironov, Kunal Talwar et Li Zhang. « R\'enyi Differential Privacy of the Sampled Gaussian Mechanism ». *arXiv preprint arXiv :1908.10530* (2019) (cf. p. 27).

[Moczulski, 2015] Marcin Moczulski, Misha Denil, Jeremy Appleyard et Nando de Freitas. « ACDC : A structured efficient linear layer ». *arXiv preprint arXiv :1511.05946* (2015) (cf. p. 37).

[Monge, 1781] Gaspard Monge. « Mémoire sur la théorie des déblais et des remblais ». *Histoire de l'Académie Royale des Sciences de Paris* (1781) (cf. p. 30).

[Montavon, 2016] Grégoire Montavon, Klaus-Robert Müller et Marco Cuturi. « Wasserstein training of restricted Boltzmann machines ». *Advances in Neural Information Processing Systems* 29 (2016), p. 3718-3726 (cf. p. 108).

[Moon, 2019a] Seungyong Moon, Gaon An et Hyun Oh Song. « Parsimonious Black-Box Adversarial Attacks via Efficient Combinatorial Optimization ». *Proceedings of the 36th International Conference on Machine Learning*. Sous la dir. de Kamalika Chaudhuri et Ruslan Salakhutdinov. T. 97. Proceedings of Machine Learning Research. Long Beach, California, USA : PMLR, 2019, p. 4636-4645 (cf. p. 24, 77, 82).

[Moon, 2019b] Seungyong Moon, Gaon An et Hyun Oh Song. « Parsimonious black-box adversarial attacks via efficient combinatorial optimization ». *arXiv preprint arXiv :1905.06635* (2019) (cf. p. 73).

[Nadjahi, 2020a] Kimia Nadjahi, Valentin De Bortoli, Alain Durmus, Roland Badeau et Umut Şimşekli. « Approximate Bayesian computation with the sliced-Wasserstein distance ». *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, p. 5470-5474 (cf. p. 109).

[Nadjahi, 2020b] Kimia Nadjahi, Alain Durmus, Lénaïc Chizat, Soheil Kolouri, Shahin Shahrampour et Umut Şimşekli. « Statistical and topological properties of sliced probability divergences ». *arXiv preprint arXiv :2003.05783* (2020) (cf. p. 109, 120, 124, 125).

[Nadjahi, 2021] Kimia Nadjahi, Alain Durmus, Pierre Jacob, Roland Badeau et Umut Simsekli. « Fast Approximation of the Sliced-Wasserstein Distance Using Concentration of Random Projections ». *Advances in Neural Information Processing Systems*. Sous la dir. d'A. Beygelzimer, Y. Dauphin, P. Liang et J. Wortman Vaughan. 2021 (cf. p. 109).

[Nadjahi, 2019] Kimia Nadjahi, Alain Durmus, Umut Şimşekli et Roland Badeau. « Asymptotic guarantees for learning generative models with the sliced-wasserstein distance ». *arXiv preprint arXiv :1906.04516* (2019) (cf. p. 109).

[Nguyen, 2021] Khai Nguyen, Nhat Ho, Tung Pham et Hung Bui. « Distributional Sliced-Wasserstein and Applications to Generative Modeling ». *International Conference on Learning Representations*. 2021 (cf. p. 109-111, 116, 117, 120, 124).

[Nøkland, 2016a] A. Nøkland. « Direct Feedback Alignment Provides Learning in Deep Neural Networks ». *Advances in Neural Information Processing Systems 29*. 2016 (cf. p. 19).

[Nøkland, 2016b] Arild Nøkland. « Direct Feedback Alignment Provides Learning in Deep Neural Networks ». *NIPS*. 2016 (cf. p. 74).

[Nøkland, 2016c] Arild Nøkland. « Direct Feedback Alignment Provides Learning in Deep Neural Networks ». *NIPS*. 2016 (cf. p. 92).

[Nøkland, 2016d] Arild Nøkland. « Direct feedback alignment provides learning in deep neural networks ». *Advances in neural information processing systems*. 2016, p. 1037-1045 (cf. p. 70).

[Oberman, 2015] Adam M Oberman et Yuanlong Ruan. « An efficient linear programming method for optimal transportation ». *arXiv preprint arXiv :1509.03668* (2015) (cf. p. 109).

[Ohana, 2021] Ruben Ohana, Hamlet Medina, Julien Launay, Alessandro Cappelli, Iacopo Poli, Liva Ralaivola et al. « Photonic Differential Privacy with Direct Feedback Alignment ». *Advances in Neural Information Processing Systems* 34 (2021) (cf. p. 5, 6, 91).

[Ohana, 2022] Ruben Ohana, Kimia Nadjahi, Alain Rakotomamonjy et Liva Ralaivola. « Shedding a PAC-Bayesian Light on Adaptive Sliced-Wasserstein Distances ». *arXiv preprint arXiv :2206.03230* (2022) (cf. p. 5, 6).

[Ohana, 2020] Ruben Ohana, Jonas Wacker, Jonathan Dong, Sébastien Marmin, Florent Krzakala, Maurizio Filippone et al. « Kernel computations from large-scale random features obtained by optical processing units ». *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, p. 9294-9298 (cf. p. 4, 6, 34, 53, 73, 95).

[Papernot, 2016] Nicolas Papernot, Patrick McDaniel et Ian Goodfellow. « Transferability in machine learning : from phenomena to black-box attacks using adversarial samples ». *arXiv preprint arXiv :1605.07277* (2016) (cf. p. 69, 73, 82).

[Papernot, 2017] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik et Ananthram Swami. « Practical black-box attacks against machine learning ». *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 2017, p. 506-519 (cf. p. 83).

[Papernot, 2018] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar et Ulfar Erlingsson. *Scalable Private Learning with PATE. arXiv preprint arXiv :1802.08908*. 2018 (cf. p. 91).

[Pardo, 2018] Leandro Pardo. *Statistical inference based on divergence measures*. CRC press, 2018 (cf. p. 96).

[Pascanu, 2013] Razvan Pascanu, Tomas Mikolov et Yoshua Bengio. « On the difficulty of training recurrent neural networks ». *International conference on machine learning*. 2013, p. 1310-1318 (cf. p. 33).

[Paszke, 2019] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan et al. « PyTorch : An Imperative Style, High-Performance Deep Learning Library ». *Advances in Neural Information Processing Systems 32*. 2019, p. 8024-8035 (cf. p. 1).

[Pathak, 2018] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu et Edward Ott. « Model-free prediction of large spatiotemporally chaotic systems from data : A reservoir computing approach ». *Physical review letters* 120.2 (2018), p. 024102 (cf. p. 34).

[Paty, 2019] François-Pierre Paty et Marco Cuturi. « Subspace robust wasserstein distances ». *International conference on machine learning*. PMLR. 2019, p. 5072-5081 (cf. p. 120).

[Peyré, 2019] Gabriel Peyré, Marco Cuturi et al. « Computational Optimal Transport : With Applications to Data Science ». *Foundations and Trends® in Machine Learning* 11.5-6 (2019), p. 355-607 (cf. p. 31, 108, 109).

[Pinot, 2019] Rafael Pinot, Laurent Meunier, Alexandre Araujo, Hisashi Kashima, Florian Yger, Cédric Gouy-Pailler et al. « Theoretical evidence for adversarial robustness through randomization : the case of the Exponential family ». *arXiv preprint arXiv :1902.01148* (2019) (cf. p. 73, 83).

[Poli, 2021] Iacopo Poli, Julien Launay, Kilian Müller, Gustave Pariente, Igor Carron, Laurent Daudet et al. *Method and system for machine learning using optical data*. 2021 (cf. p. 7, 22).

[Powell, 2008] James R Powell. « The quantum limit to Moore's law ». *Proceedings of the IEEE* 96.8 (2008), p. 1247-1248 (cf. p. 1).

[Rabin, 2012] Julien Rabin, Gabriel Peyré, Julie Delon et Marc Bernot. « Wasserstein Barycenter and Its Application to Texture Mixing ». *Scale Space and Variational Methods in Computer Vision*. Sous la dir. d'Alfred M. Bruckstein, Bart M. ter Haar Romeny, Alexander M. Bronstein et Michael M. Bronstein. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, p. 435-446 (cf. p. 32, 109).

[Radford, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever et al. « Language models are unsupervised multitask learners ». *OpenAI blog* 1.8 (2019), p. 9 (cf. p. 2).

[Rahimi, 2007] Ali Rahimi et Benjamin Recht. « Random Features for Large-Scale Kernel Machines ». *Advances in Neural Information Processing Systems 20*. Curran Associates Inc., 2007, p. 1177-1184 (cf. p. 14).

[Rahimi, 2008] Ali Rahimi et Benjamin Recht. « Random features for large-scale kernel machines ». *Advances in neural information processing systems*. 2008, p. 1177-1184 (cf. p. 11, 34, 36, 54).

[Rahimi, 2009] Ali Rahimi et Benjamin Recht. « Weighted sums of random kitchen sinks : Replacing minimization with randomization in learning ». *Advances in neural information processing systems*. 2009, p. 1313-1320 (cf. p. 34, 54).

[Rakotomamonjy, 2021] Alain Rakotomamonjy et Liva Ralaivola. « Differentially private sliced wasserstein distance ». *International Conference on Machine Learning*. PMLR. 2021, p. 8810-8820 (cf. p. 109).

[Ramesh, 2022] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu et Mark Chen. « Hierarchical text-conditional image generation with clip latents ». *arXiv preprint arXiv :2204.06125* (2022) (cf. p. 2).

[Ramesh, 2021] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford et al. « Zero-shot text-to-image generation ». *International Conference on Machine Learning*. PMLR. 2021, p. 8821-8831 (cf. p. 2).

[Refinetti, 2021] Maria Refinetti, Stéphane D'Ascoli, Ruben Ohana et Sebastian Goldt. « Align, then memorise : the dynamics of learning with feedback alignment ». *Proceedings of the 38th International Conference on Machine Learning*. Sous la dir. de Marina Meila et Tong Zhang. T. 139. Proceedings of Machine Learning Research. PMLR, 2021, p. 8925-8935 (cf. p. 5, 6, 17, 18, 20, 21, 75, 93).

[Ren, 2019] Kui Ren, Qian Wang, Cong Wang, Zhan Qin et Xiaodong Lin. « The security of autonomous driving : Threats, defenses, and future directions ». *Proceedings of the IEEE* 108.2 (2019), p. 357-372 (cf. p. 24).

[Rényi, 1961] Alfréd Rényi. « On Measures of Entropy and Information ». *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1 : Contributions to the Theory of Statistics*. Berkeley, Calif. : University of California Press, 1961, p. 547-561 (cf. p. 27, 95).

[Rotskoff, 2018] G.M. Rotskoff et E. Vanden-Eijnden. « Parameters as interacting particles : long time convergence and asymptotic error scaling of neural networks ». *Advances in Neural Information Processing Systems 31*. 2018, p. 7146-7155 (cf. p. 19).

[Roy, 2020] Deboleena Roy, Indranil Chakraborty, Timur Ibrayev et Kaushik Roy. « Robustness Hidden in Plain Sight : Can Analog Computing Defend Against Adversarial Attacks ? » *arXiv preprint arXiv :2008.12016* (2020) (cf. p. 83).

[Rudi, 2015] Alessandro Rudi, Raffaello Camoriano et Lorenzo Rosasco. « Less is More : Nyström Computational Regularization ». *Advances in Neural Information Processing Systems 28*. Sous la dir. de C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama et R. Garnett. Curran Associates, Inc., 2015, p. 1657-1665 (cf. p. 53).

[Rudi, 2017a] Alessandro Rudi, Luigi Carratino et Lorenzo Rosasco. « Falkon : An optimal large scale kernel method ». *Advances in Neural Information Processing Systems*. 2017, p. 3888-3898 (cf. p. 53, 57).

[Rudi, 2017b] Alessandro Rudi et Lorenzo Rosasco. « Generalization properties of learning with random features ». *Advances in Neural Information Processing Systems*. 2017, p. 3215-3225 (cf. p. 34, 38).

[Rumelhart, 1986] David E. Rumelhart, Geoffrey E. Hinton et Ronald J. Williams. « Learning representations by back-propagating errors ». *Nature* 323.6088 (1986), p. 533-536 (cf. p. 17, 93).

[Russakovsky, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma et al. « ImageNet Large Scale Visual Recognition Challenge ». *International Journal of Computer Vision (IJCV)* 115.3 (2015), p. 211-252 (cf. p. 58).

[Saad, 2009] D. Saad. *On-line learning in neural networks*. T. 17. Cambridge University Press, 2009 (cf. p. 19).

[Saad, 1995a] D. Saad et S.A. Solla. « Exact Solution for On-Line Learning in Multilayer Neural Networks ». *Phys. Rev. Lett.* 74.21 (1995), p. 4337-4340 (cf. p. 19, 20).

[Saad, 1995b] D. Saad et S.A. Solla. « On-line learning in soft committee machines ». *Phys. Rev. E* 52.4 (1995), p. 4225-4243 (cf. p. 19, 20).

[Saade, 2016a] A. Saade, F. Caltagirone, I. Carron, L. Daudet, A. Drémeau, S. Gigan et al. « Random projections through multiple optical scattering : Approximating Kernels at the speed of light ». *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, p. 6215-6219 (cf. p. 54, 55).

[Saade, 2016b] Alaa Saade, Francesco Caltagirone, Igor Carron, Laurent Daudet, Angélique Drémeau, Sylvain Gigan et al. « Random projections through multiple optical scattering : Approximating kernels at the speed of light ». *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, p. 6215-6219 (cf. p. 34).

[Salehinejad, 2017] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak et Shahrokh Valaee. « Recent advances in recurrent neural networks ». *arXiv preprint arXiv :1801.01078* (2017) (cf. p. 33).

[Salman, 2020] Hadi Salman, Andrew Ilyas, L. Engstrom, Ashish Kapoor et A. Madry. « Do Adversarially Robust ImageNet Models Transfer Better ? » *ArXiv* abs/2007.08489 (2020) (cf. p. 83).

[Saxe, 2018] A.M. Saxe, Y. Bansal, J. Dapello, M.S. Advani, A. Kolchinsky, B.D. Tracey et al. « On the information bottleneck theory of deep learning ». *ICLR*. 2018 (cf. p. 19).

[Schmitz, 2018] Morgan A Schmitz, Matthieu Heitz, Nicolas Bonneel, Fred Ngole, David Coeurjolly, Marco Cuturi et al. « Wasserstein dictionary learning : Optimal transport-based unsupervised nonlinear dictionary learning ». *SIAM Journal on Imaging Sciences* 11.1 (2018), p. 643-678 (cf. p. 108).

[Schmitzer, 2016] Bernhard Schmitzer. « A sparse multiscale algorithm for dense optimal transport ». *Journal of Mathematical Imaging and Vision* 56.2 (2016), p. 238-259 (cf. p. 109).

[Scholkopf, 2002] Bernhard Scholkopf et Alexander J. Smola. *Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002 (cf. p. 10, 11).

[Schölkopf, 2002] Bernhard Schölkopf, Alexander J Smola, Francis Bach et al. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT press, 2002 (cf. p. 10, 53, 57).

[Schrauwen, 2009] Benjamin Schrauwen, Lars Büsing et Robert A Legenstein. « On computational power and the order-chaos phase transition in reservoir computing ». *Advances in Neural Information Processing Systems*. 2009, p. 1425-1432 (cf. p. 41).

[Scott, 2021] T. R. Scott, A. C. Gallagher et M. C. Mozer. « von Mises–Fisher Loss : An Exploration of Embedding Geometries for Supervised Learning ». *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA : IEEE Computer Society, 2021, p. 10592-10602 (cf. p. 117).

[Sehwag, 2021] V. Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, M. Chiang et al. « Improving Adversarial Robustness Using Proxy Distributions ». *ArXiv* abs/2104.09425 (2021) (cf. p. 85, 86).

[Seung, 1992] H. S. Seung, H. Sompolinsky et N. Tishby. « Statistical mechanics of learning from examples ». *Physical Review A* 45.8 (1992), p. 6056-6091 (cf. p. 19).

[Shafahi, 2020] A. Shafahi, Parsa Saadatpanah, C. Zhu, Amin Ghiasi, C. Studer, D. Jacobs et al. « Adversarially robust transfer learning ». *ArXiv* abs/1905.08232 (2020) (cf. p. 83).

[Silver, 2016] D. Silver, Aja Huang, Chris J. Maddison, A. Guez, Laurent Sifre, George van den Driessche et al. « Mastering the game of Go with deep neural networks and tree search ». *Nature* 529.7587 (2016), p. 484-489 (cf. p. 2).

[Silver, 2018] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez et al. « A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play ». *Science* 362.6419 (2018), p. 1140-1144 (cf. p. 2).

[Simonyan, 2014a] K. Simonyan et A. Zisserman. « Very Deep Convolutional Networks for Large-Scale Image Recognition ». *CoRR* abs/1409.1556 (2014) (cf. p. 58).

[Simonyan, 2014b] Karen Simonyan et Andrew Zisserman. « Very deep convolutional networks for large-scale image recognition ». *arXiv preprint arXiv :1409.1556* (2014) (cf. p. 75).

[Sirignano, 2019] J. Sirignano et K. Spiliopoulos. « Mean field analysis of neural networks : A central limit theorem ». *Stochastic Processes and their Applications* (2019) (cf. p. 19).

[Sivashinsky, 1977] GI Sivashinsky. « Nonlinear analysis of hydrodynamic instability in laminar flames—I. Derivation of basic equations ». *Acta astronautica* 4 (1977), p. 1177-1206 (cf. p. 42).

[Smola, 2000] Alex J Smola et Bernhard Schölkopf. « Sparse greedy matrix approximation for machine learning » (2000) (cf. p. 53).

[Solomon, 2014] Justin Solomon, Raif Rustamov, Leonidas Guibas et Adrian Butscher. « Wasserstein propagation for semi-supervised learning ». *International Conference on Machine Learning*. PMLR. 2014, p. 306-314 (cf. p. 108).

[Soltanolkotabi, 2018] M. Soltanolkotabi, A. Javanmard et J.D. Lee. « Theoretical insights into the optimization landscape of over-parameterized shallow neural networks ». *IEEE Transactions on Information Theory* 65.2 (2018), p. 742-769 (cf. p. 19).

[Tanaka, 2019] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda et al. « Recent advances in physical reservoir computing : A review ». *Neural Networks* (2019) (cf. p. 34).

[Thomas, 2018] Anna Thomas, Albert Gu, Tri Dao, Atri Rudra et Christopher Ré. « Learning compressed transforms with low displacement rank ». *Advances in neural information processing systems*. 2018, p. 9052-9060 (cf. p. 37).

[Tian, 2017] Y. Tian. « An Analytical Formula of Population Gradient for Two-Layered ReLU Network and Its Applications in Convergence and Critical Point Analysis ». *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, p. 3404-3413 (cf. p. 19).

[Tolstikhin, 2017] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly et Bernhard Schoelkopf. « Wasserstein auto-encoders ». *arXiv preprint arXiv :1711.01558* (2017) (cf. p. 108).

[Tong, 2018] Zhiqiang Tong et Gouhei Tanaka. « Reservoir computing with untrained convolutional neural networks for image recognition ». *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE. 2018, p. 1289-1294 (cf. p. 34).

[Tramer, 2020] Florian Tramer, Nicholas Carlini, Wieland Brendel et Aleksander Madry. *On Adaptive Attacks to Adversarial Example Defenses. arXiv preprint arXiv :2002.08347*. 2020 (cf. p. 73).

[Tramèr, 2019] Florian Tramèr et Dan Boneh. « Adversarial training and robustness for multiple perturbations ». *Advances in Neural Information Processing Systems*. 2019, p. 5866-5876 (cf. p. 70, 83).

[Tsipras, 2019] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner et Aleksander Madry. « Robustness May Be at Odds with Accuracy ». *International Conference on Learning Representation* (2019) (cf. p. 70).

[Van der Sande,] Guy Van der Sande, Daniel Brunner et Miguel C Soriano. « Advances in photonic reservoir computing ». *Nanophotonics* 6.3 (), p. 561-576 (cf. p. 34).

[Vaswani, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez et al. « Attention is all you need ». *Advances in neural information processing systems*. 2017, p. 5998-6008 (cf. p. 17).

[Vershynin, 2018] Roman Vershynin. *High-Dimensional Probability : An Introduction with Applications in Data Science*. Cambridge University Press, 2018 (cf. p. 64).

[Verstraeten, 2007] David Verstraeten, Benjamin Schrauwen, Michiel d'Haene et Dirk Stroobandt. « An experimental unification of reservoir computing methods ». *Neural networks* 20.3 (2007), p. 391-403 (cf. p. 33).

[Villani, 2008] Cédric Villani. *Optimal transport : old and new*. T. 338. Springer Science & Business Media, 2008 (cf. p. 31).

[Villani, 2009] Cédric Villani. *Optimal transport : old and new*. T. 338. Springer, 2009 (cf. p. 108).

[Vlachas, 2019] Pantelis R Vlachas, Jaideep Pathak, Brian R Hunt, Themistoklis P Sapsis, Michelle Girvan, Edward Ott et al. « Forecasting of spatio-temporal chaotic dynamics with recurrent neural networks : A comparative study of reservoir computing and backpropagation algorithms ». *arXiv preprint arXiv :1910.05266* (2019) (cf. p. 42).

[Wacker, 2022a] Jonas Wacker, Motonobu Kanagawa et Maurizio Filippone. « Improved Random Features for Dot Product Kernels ». *arXiv preprint arXiv :2201.08712* (2022) (cf. p. 5).

[Wacker, 2022b] Jonas Wacker, Ruben Ohana et Maurizio Filippone. « Complex-to-Real Random Features for Polynomial Kernels ». *arXiv preprint arXiv :2202.02031* (2022) (cf. p. 7, 14-16).

[Wainrib, 2016] Gilles Wainrib et Mathieu N Galtier. « A local echo state property through the largest Lyapunov exponent ». *Neural Networks* 76 (2016), p. 39-45 (cf. p. 41).

[Wang, 2015] Qian Wang, Yingyezhe Jin et Peng Li. « General-purpose LSM learning processor architecture and theoretically guided design space exploration ». *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE. 2015, p. 1-4 (cf. p. 34).

[Wang, 2019] Yu-Xiang Wang, Borja Balle et Shiva Prasad Kasiviswanathan. « Subsampled Rényi differential privacy and analytical moments accountant ». *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, p. 1226-1235 (cf. p. 27, 96, 100).

[Watkin, 1993] T.L.H. Watkin, A. Rau et M. Biehl. « The statistical mechanics of learning a rule ». *Reviews of Modern Physics* 65.2 (1993), p. 499-556 (cf. p. 19).

[Weed, 2019] Jonathan Weed et Francis Bach. « Sharp asymptotic and finite-sample rates of convergence of empirical measures in Wasserstein distance ». *Bernoulli* 25.4 A (2019), p. 2620-2648 (cf. p. 114, 123).

[Wetzstein, 2020] Gordon Wetzstein, Aydogan Ozcan, Sylvain Gigan, Shanhui Fan, Dirk Englund, Marin Soljačić et al. « Inference in artificial intelligence with deep optics and photonics ». *Nature* 588.7836 (2020), p. 39-47 (cf. p. 102).

[Wierstra, 2014] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters et Jürgen Schmidhuber. « Natural evolution strategies ». *The Journal of Machine Learning Research* 15.1 (2014), p. 949-980 (cf. p. 24).

[Wong, 2018a] Eric Wong et Zico Kolter. « Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope ». *International Conference on Machine Learning*. 2018, p. 5286-5295 (cf. p. 73, 83).

[Wong, 2020] Eric Wong, Leslie Rice et J. Z. Kolter. « Fast is better than free : Revisiting adversarial training ». *ArXiv* abs/2001.03994 (2020) (cf. p. 85, 86).

[Wong, 2018b] Eric Wong, Frank Schmidt, Jan Hendrik Metzen et J Zico Kolter. « Scaling provable adversarial defenses ». *Advances in Neural Information Processing Systems*. 2018, p. 8400-8409 (cf. p. 73, 83).

[Wu, 2020] Dongxian Wu, Shutao Xia et Yisen Wang. « Adversarial Weight Perturbation Helps Robust Generalization ». *arXiv : Learning* (2020) (cf. p. 85, 86).

[Xiao, 2017] Han Xiao, Kashif Rasul et Roland Vollgraf. « Fashion-mnist : a novel image dataset for benchmarking machine learning algorithms ». *arXiv preprint arXiv :1708.07747* (2017) (cf. p. 102, 119).

[Xu, 2020] Depeng Xu, Wei Du et Xintao Wu. « Removing Disparate Impact of Differentially Private Stochastic Gradient Descent on Model Accuracy ». *arXiv preprint arXiv :2003.03699* (2020) (cf. p. 103).

[Xu, 2021] Xingyuan Xu, Mengxi Tan, Bill Corcoran, Jiayang Wu, Andreas Boes, Thach G Nguyen et al. « 11 TOPS photonic convolutional accelerator for optical neural networks ». *Nature* 589.7840 (2021), p. 44-51 (cf. p. 102).

[Yamaguchi, 2006] Ichirou Yamaguchi. « Phase-shifting digital holography ». *Digital Holography and Three-Dimensional Display*. Springer, 2006, p. 145-171 (cf. p. 22).

[Yoshida, 2019] Y. Yoshida et M. Okada. « Data-Dependence of Plateau Phenomenon in Learning with Neural Network — Statistical Mechanical Analysis ». *Advances in Neural Information Processing Systems 32*. 2019, p. 1720-1728 (cf. p. 19).

[Yu, 2016] Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice et Sanjiv Kumar. « Orthogonal random features ». *Advances in Neural Information Processing Systems*. 2016, p. 1975-1983 (cf. p. 34, 37).

[Zantedeschi, 2021] Valentina Zantedeschi, Paul Viallard, Emilie Morvant, Rémi Emonet, Amaury Habrard, Pascal Germain et al. « Learning Stochastic Majority Votes by Minimizing a PAC-Bayes Generalization Bound ». *NeurIPS*. Online, France, 2021 (cf. p. 109).

[Zdeborová, 2016] L. Zdeborová et F. Krzakala. « Statistical physics of inference : thresholds and algorithms ». *Adv. Phys.* 65.5 (2016), p. 453-552 (cf. p. 19).

[Zhang, 2020] Jingfeng Zhang, Jianing Zhu, Gang Niu, B. Han, M. Sugiyama et M. Kankanhalli. « Geometry-aware Instance-reweighted Adversarial Training ». *ArXiv* abs/2010.01736 (2020) (cf. p. 85, 86).

[Zhang, 2015] Yong Zhang, Peng Li, Yingyezhe Jin et Yoonsuck Choe. « A digital liquid state machine with biologically inspired learning and its application to speech recognition ». *IEEE transactions on neural networks and learning systems* 26.11 (2015), p. 2635-2649 (cf. p. 34).

[Zhang, 2013] Yuchen Zhang, John Duchi et Martin Wainwright. « Divide and conquer kernel ridge regression ». *Conference on Learning Theory*. 2013, p. 592-617 (cf. p. 53).

[Zhong, 2017] K. Zhong, Z. Song, P. Jain, P.L. Bartlett et I.S. Dhillon. « Recovery guarantees for one-hidden-layer neural networks ». *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 2017, p. 4140-4149 (cf. p. 19).

RÉSUMÉ

Dans cette thèse, nous tirerons parti de l'usage de l'aléatoire dans différents aspects de l'apprentissage automatique. Nous commençerons par montrer le lien entre le calcul par réservoir et les noyaux récurrents sous le prisme des caractéristiques aléatoires, et introduirons les transformées structurées afin d'améliorer la complexité computationnelle du calcul par réservoir. Par la suite, nous montrerons comment tirer parti de calculs optiques afin de mettre à l'échelle les caractéristiques aléatoires pour l'approximation de noyaux, à bas coût énergétique. Nous continuerons par montrer comment combiner le Processeur de Calcul Optique avec des méthodes d'entraînement alternatives à la rétropropagation du gradient tel que l'alignement par retour direct, afin de rendre adversariallement robuste des réseaux de neurones entraînés depuis le début ou d'augmenter la robustesse des défenses les plus robustes. Par ailleurs, nous entraînerons un réseau de neurones de façon optique et tirerons parti du bruit expérimental afin de démontrer comment cela induit une confidentialité différentielle. Nous finirons par utiliser les bornes PAC-Bayésiennes afin d'optimiser la distribution des projections aléatoires de la distance de Sliced-Wasserstein, tout en s'appuyant sur des fondations théoriques.

MOTS CLÉS

caractéristiques aléatoires, calcul optique, méthodes à noyau, calcul à réservoir, retour par alignement direct, robustesse adversarialle, confidentialité différentielle, transport optimal.

ABSTRACT

In this thesis, we will leverage the use of randomness in multiple aspects of machine learning. We will start by showing the link between reservoir computing and recurrent kernels through the lens of random features, and introduce structured transforms to improve the computational complexity of reservoir computing. We will then show how optical computing can help scaling-up random features for kernel approximation, at a low energy cost. We will continue by showing how to combine the Optical Processing Unit with training methods alternative to backpropagation such as Direct Feedback Alignment, to make adversarially robust networks trained from the beginning, or improve the robustness of state-of-the-art defenses. We will also train optically a neural network and show how the experimental noise yields differential privacy. We will finish by using PAC-Bayesian bounds to optimize the distribution of random projections of Sliced-Wasserstein distances while being theoretically grounded.

KEYWORDS

random features, optical computing, kernel methods, reservoir computing, direct feedback alignment, adversarial robustness, differential privacy, optimal transport.