# Contributions to Local, Asynchronous and Decentralized Learning, and to Geometric Deep Learning

Edouard Oyallon

▶ **To cite this version:**

Edouard Oyallon. Contributions to Local, Asynchronous and Decentralized Learning, and to Geometric Deep Learning. Artificial Intelligence [cs.AI]. Sorbonne Université, 2023. tel-04334118

**HAL Id: tel-04334118**
**https://hal.science/tel-04334118**

Submitted on 10 Dec 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

SCIENCES SORBONNE UNIVERSITÉ

# MÉMOIRE D'HABILITATION À DIRIGER DES RECHERCHES

Opéré au sein de :
**Sorbonne Université**

**Spécialité :** Informatique

Soutenu publiquement le 08/12/2023, par :

**Edouard Oyallon**

---

# Contributions to Local, Asynchronous and Decentralized Learning, and to Geometric Deep Learning

---

Devant le jury composé de :

| | |
|---|---|
| **Jamal ATIF**<br>Professeur des Universités, Université Paris Dauphine-PSL | **Examinateur** |
| **Émilie CHOUZENOUX**<br>Directrice de Recherche, Inria | **Rapporteuse** |
| **Patrick GALLINARI**<br>Professeur des Universités, Sorbonne Université/Criteo | **Président du Jury** |
| **Julien MAIRAL**<br>Directeur de Recherche, Inria | **Rapporteur** |
| **Sebastian STICH**<br>Chercheur, CISPA Helmholtz Center for Information Security | **Rapporteur** |
| **Michal VALKO**<br>Chargé de Recherche, Inria/Google Deepmind | **Examinateur** |

# Contents

This work was supported by the ANR JCJC ADONIS ANR-21-CE23-0030.

# Remerciements

Tout d'abord, et contrairement à l'un des gros oublis de ma thèse, j'aimerai remercier les rapporteurs de ce manuscrit car je commence à réaliser la pénibilité de certaines tâches rébarbatives du métier de chercheur. Ensuite, les autres membres de mon jury, et en particulier Michal, dont les conseils m'ont beaucoup éclairé depuis la fin de ma thèse. Merci aussi à MLIA de m'avoir accueilli dans un environnement où j'ai beaucoup de liberté intellectuelle. Je remercie ensuite l'ensemble des collaborateurs qui ont eu l'envie d'écrire des articles avec moi, et qui ont contribué à l'avancée de ma pensée de chercheur. Je remercie enfin mes étudiants de me suivre et d'élaguer avec moi les pistes que je pense intéressantes, et de me suivre dans nos erreurs comme dans nos succès.

Ensuite, j'aimerais remercier mes proches pour leur soutien, qui ont beaucoup entendu parler de ce métier-passion très particulier. Je vous rassure, c'est mon dernier diplôme. :-)

À vous tous, et aux prochaines aventures.

# Résumé

Ce document est un résumé d'une partie des recherches que j'ai menées de 2018 à 2023 afin d'obtenir le diplôme d'Habilitation à Diriger des Recherches. L'ensemble des résultats mentionnés est discuté plus en profondeur dans les publications correspondantes.

Les sujets des recherches dont je vais discuter portent sur les notions d'apprentissage distribué, d'apprentissage local et d'apprentissage de représentation profonde sur des variétés. Un des points central que je cherche à développer est la création d'algorithme distribué d'entrainement de réseau de neurones profond, efficace, tant du point de vu du temps de calcul que de la quantité d'opération nécessaire pour l'entrainement. Le premier chapitre correspond à une introduction à ces domaines ainsi qu'à mes activités de recherche. Le second chapitre discute des relations entre le matériel informatique et les paradigmes existants d'entrainement. Ces interactions sont importantes car elles permettent de promettre ou d'obtenir des améliorations significatives des coûts d'entrainement des réseaux profonds. Le troisième chapitre décrit des éléments initiaux d'une recherche pour créer des représentations profondes sur variété et graphe. En particulier, nous avons essayé de réfléchir à des principes pour modéliser de telles architectures. Le quatrière chapitre discute des résultats en apprentissage local que j'ai obtenu, et notamment en apprentissage glouton couche par couche. Cela correspond à l'idée d'entraîner des réseaux en proposant des mises à jour des poids d'une couche basées uniquement sur des informations locales à cette couche, et en utilisant le moins possible d'information globale liée à toutes les couches. Le cinquième chapitre résume des résultats obtenus en apprentissage asynchrone décentralisé, dans des cadres d'optimisation convexe et d'optimisation de poids de réseaux de neurones profonds. Un des intérêts de ce type d'approche est leur potentielle supériorité, tant d'un point de vue d'implémentation pratique que d'un point de vue algorithmique. Enfin, le dernier chapitre propose des perspectives futures sur ma recherche.

# Summary

This document is a summary of some of the research I conducted from 2018 to 2023 to obtain the *Habilitation à Diriger des Recherches*. All the results mentioned are discussed in greater depth in the corresponding publications.

The research topics I will discuss focus on the concepts of distributed learning, local learning, and deep representation learning on manifolds. A central point I aim to develop is the creation of efficient distributed algorithms for training deep neural networks, both in terms of computation time and the number of operations required for training. The first chapter provides an introduction to these fields and my research activities. The second chapter discusses the interplay between computer hardware and existing training paradigms. These interactions are crucial as they promise or achieve significant reductions in the training costs of deep networks. The third chapter outlines initial elements of research to create deep representations on manifolds and graphs. In particular, we have sought principles for modeling such architectures. The fourth chapter discusses the local learning results I have obtained, especially layer-by-layer greedy learning. This corresponds to the idea of training networks by proposing weight updates for a layer based solely on local information to that layer, using as little global information related to all layers as possible. The fifth chapter summarizes results obtained in decentralized asynchronous learning, within frameworks of convex optimization and optimization of deep neural network weights. One of the interests of such approaches is their potential superiority, both from a practical implementation standpoint and from an algorithmic perspective. Finally, the last chapter provides future perspectives on my research.

# Chapter 1

# Introduction

Deep Learning has revolutionized various scientific fields in under a decade. When I embarked on my PhD journey in 2013 on "*Learning invariants for Image Classification*" whose goal was to understand the Deep Learning black-box machinery, I initially viewed Deep Learning as merely another tool to address Machine Learning challenges. However, its incomparable flexibility, efficiency, and generalization capabilities—unseen in other paradigms—soon positioned it as a go-to solution. Today, the research landscape around Deep Learning is vast, teeming with more questions than answers regarding the inner workings of these models. Instead of delving deep to understand them, many researchers are making their routine algorithms differentiable and integrating them with Deep Learning. This approach often sidesteps theoretical considerations but consistently delivers enhanced performance. Observing this trend prompted me to shift my research focus in 2020. My primary aim now is to devise innovative, task-agnostic algorithms that enable faster, larger, deeper, and more efficient training of Deep Neural Networks, broadening their potential applications.

## 1.1 Scientific context

Training Deep Neural Networks demands considerable computational resources owing to the size and complexity of modern models and the extensive amounts of data required for training [Krizhevsky et al., 2012]. Researchers have explored various parallelization techniques to expedite the training process and manage these challenges. The primary focus has been on two complementary and concurrent strategies: model parallelism [Shoeybi et al., 2019] and data parallelism [Subhlok et al., 1993] (see Figure 1.1). These strategies enable efficient use of multiple processors or computing nodes to distribute the workload and enhance Deep Neural Network training speed. In this manuscript, I intend to concentrate on algorithmic solutions to the following hardware bottlenecks, aimed at reducing:

- Total number of computations,

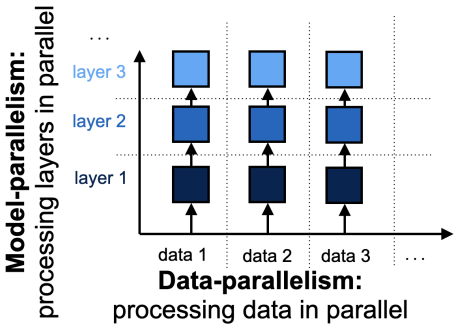


Figure 1.1: Illustration of the difference of paradigm between model and data parallelism.

- Overall communication volume,
- Local bandwidth usage,
- Overall training time,

while avoiding any degradation in final accuracy or performance. This last point is the most challenging, as any attempt to modify standard Deep Learning pipelines requires to run a a large collection of challenging benchmarks. Note that all those elements have an impact on the ecological footprint of Deep Neural Network training [Patterson et al., 2022].

**Beyond parallel computing for training Deep Neural Networks: distributed algorithms for Deep Neural Network training.** Parallel computing [Atre et al., 2021] represents the current standard paradigm for designing and structuring Deep Neural Network training pipelines. Typically implemented via either pipeline-parallelism or data-parallelism, this approach essentially considers the standard Stochastic Gradient Descent (SGD) training algorithm and segments the training data into smaller chunks that can be processed in parallel (potentially, by modules in parallel, leading to an extra model-parallelism) and results are aggregated in a shared memory. This paradigm exemplifies a parallel algorithm, as it necessitates the use of a global orchestrator (e.g., a node or an implicit set of instruction such as wait barriers) to concurrently execute a sequential list of instructions, resulting in a gradient step.

While this enables parallelization of computations [Dean and Ghemawat, 2008], this training paradigm often remains unchallenged. Rather than focusing on innovative training methods, most efforts to speed up training involve using larger GPUs connected with high-bandwidth links, enabling nearly instantaneous computation. This allows the cluster to function as a synchronous machine, which is appealing but limits the size, hardware, and topology of the utilized network [Sze et al., 2017]. This limit is physical [Milton and Zarkesh-Ha, 2023]: it means that it can be ignored up to a certain point and pushed

away, yet it will always come back when requiring larger machines. Moreover, by its very nature, a Deep Neural Network model is a sequential composition of layers with distinct purposes and potentially different convergence properties, which is not well-aligned with parallel-computing approaches.

Distributed computing [Bal et al., 1989] represents a broader paradigm for developing training algorithms: instead of assuming shared memory, it posits that each computation can be performed on a local node with its own memory. This lack of shared memory removes communication and bandwidth bottleneck [Campbell et al., 1983], that would inevitably surface beyond a certain scale in parallel computing paradigm. Parallel computing, therefore, can be understood as a specific instance of distributed computing, whereas distributed computing can operate in much larger-scale settings as these algorithms can be designed to work on large scale environments where latency, communications, and heterogeneity are key factors [Dean et al., 2012]. Neither of these approaches takes into account the heterogeneous nature of Deep Neural Network layers or minor hardware fluctuations that could make distributed computing more efficient, i.e., an approach that segments the problem into smaller, distinct instances that do not require constant communication or synchronization.

**Removing global orchestrator.** To be truly effective and allow parallel computations, distributed computing must eliminate any global orchestration while maintaining high final accuracy. This implies that there should be no wait barriers and at the same time, no performance degradation. This leads us to the concept of decoupled computations and communications, where communication or computations need not be synchronous or executed in a specific order. For modeling the independent behavior of each computational unit, the notion of decentralized algorithm becomes important. This approach is inherently asynchronous and, therefore, does not require any central units, as no shared memory would exist. Although the advantage of asynchrony is the absence of synchronization steps, it can also make the procedure more unstable, posing a challenge for Deep Neural Network training, which my research is attacking.

Moving forward, we will model asynchrony, communications, and computations within a common framework inspired by Even et al. [2021]. We assume the existence of $n$ nodes, each with local memory capable of performing computations specific to node $i$. The topology guiding the order of communications is given by potentially time-varying graphs $\mathcal{E}(t)$ where $t \in \mathbb{R}_+$ is a (often continuous) time index. Our goal is to study the parameter dynamic $x_i(t)$, which evolves according to the following delayed equations

$$dx_i(t) = \text{computations}_i(x_i(t - \tau_i(t)))dM_i(t) \tag{1.1}$$

$$+ \sum_{j,(ij)\in\mathcal{E}(t)} \text{communications}_{ij}(x_i(t - _{\tau ij}\,(t)), x_j(t - \tau_{ij}(t)))dN_{ij}(t)\,. \tag{1.2}$$

Here, $M_i, N_{ij}$ are pointwise processes (either deterministic or stochastic) that

depend upon time, and $\tau_i, \tau_{ij}$ model delays due to machine lag or distance between workers. I would like to emphasize that $x_i$ does not necessarily represent the parameters of the whole model: in fact it could correspond to the parameter of a specific module or to individual slice of parameters. For now, we consider computations$_i$ and computations$_{ij}$ as black-box operations encapsulating any kind of operations mixing, that would manipulate and communicate those slice of parameters $x_i$ or pair of parameters $\{x_i, x_j\}$. This framework can accommodate back-propagation (using a line graph aligned with the feedforward nature of the Deep Neural Network), layer-wise communication and synchronous communications as well as mini-batch SGD computations. Thus, this extremely general setup will form the foundational framework of this manuscript and my research.

Asynchrony in optimization offers not only the aforementioned practical advantages but also theoretical superiority compared to synchronous algorithms: this is because asynchrony allows a notion of randomness or pseudo-randomness [Even et al., 2021]. At a first approximation, it is reasonable to assume that $M_i, N_{ij}$ are independent stochastic processes. In this case, several works [Defazio et al., 2014, Woodworth and Srebro, 2017] have observed that an additional theoretical acceleration can be achieved, compared to the synchronous deterministic setting. However, it should be noted that, at the time of writing and to my knowledge, no one has theoretically reached the optimal rates for strongly and smooth convex functions. One of my belief is that this advantage of asynchrony can transpose to Deep Neural Network training, in the specific setting of primal methods (there is no notion of duality for Deep Neural Networks). However, for obtaining asynchronous algorithms, it is necessary to "break" any synchronous barrier and instructions of a training algorithm: in other words, we need to be able to process independent part of the network in parallel and to effectively handle how those parts interact.

For increasing potential parallelization, Machine Learning offers two paradigms (see Figure 1.1), that can be potentially combined together:

- Model-parallelism [Shoeybi et al., 2019]: the principle of splitting a Deep Neural Network model into subsequent layers, which can be processed in parallel.
- Data-parallelism [Subhlok et al., 1993]: the principle of splitting a data in batches, which can be processed in parallel.

In both case, this requires to have additional buffer variables and communications for:

- Communicating and storing features between subsequent layers (which can be highly-dimensional) with other clients,
- Communicating and storing parameters of copy of layers of a given depth.

**Intricate Intersections of Graph Concepts** Decentralized learning inherently intertwines with graph theory, as graphs serve as natural models for the

interactions among nodes [Boyd et al., 2006b]. One captivating aspect of this overlap is the concept of graph resistance [Klein and Randić, 1993], which provides a straightforward yet powerful framework for understanding gossip algorithms. This connection has inspired me to integrate my work on manifold learning into this manuscript. Essentially, manifolds can be approximated as graphs [Hammond et al., 2011, Coifman et al., 2005, Bronstein et al., 2021], establishing a seamless and intellectually stimulating link between insights for manifold learning and decentralized training methods.

## 1.2 Research Activity

I will now enumerate some of my scientific contributions since the inception of my research career. Certain publications do not neatly fit into any of the aforementioned categories. For instance, my first publication was in the journal IPOL [Oyallon and Rabin, 2015] and it remains a pivotal milestone for which I owe immense gratitude to Prof. Jean-Michel Morel and Prof. Julien Rabin for setting me on my research journey.

### 1.2.1 Contributions not discussed in this manuscript

I will begin by discussing my work on the Scattering Transform, which can be understood as a predefined Neural Network with mathematical guarantees. This framework offers considerable flexibility for working with various structures, including manifolds [Gama et al., 2019].

**Scattering Transform: an empirical contribution during my Ph.D.** During my PhD which started in 2012 (see [Oyallon, 2017a]), I had the opportunity to work on the Scattering Transform [Bruna and Mallat, 2013, Mallat, 2012, 2010]. Taking foundations in signal processing, it is a non-linear operator which consists in a cascade of point-wise modulus non-linearity and wavelet transform [Mallat, 1999]. The main principle of the Scattering Transform is to create invariance w.r.t. a group of geometric variabilities: the two major examples being the finite group of translation and the non-compact group of diffeomorphism. In this context, I have mainly worked from the image processing perspective: image classification [Oyallon et al., 2013, Oyallon and Mallat, 2015, Oyallon et al., 2017], image detection [Oyallon, 2016], for which geometric variabilities are naturally embedded in the Euclidean group. One of my main contribution to this area is the notion of hybrid networks, which mix predefined Scattering Transforms with End-to-End learned Deep Neural Networks. Indeed, the *a priori* Incorporated in the Scattering Transform transfer favorably to the Deep Neural Networks, allowing for instance more stable Deep Neural Networks, an improvement in limited labeled data settings or a small computation gain by not requiring early layers learning via back-propagation.

This has been possible thanks to the development of Kymat.io [Andreux et al., 2020], for which I am one of the main contributor. This is a library which

is commonly used as a standard implementation of the Scattering Transform, and which results from an international collaboration. It can be used to solve a wide range of problems, from inverse problems of the Scattering coefficients [Han and Lostanlen, 2020], to image detection [Oyallon et al., 2017].

The Scattering Transform has allowed me to ask a nice, simple question: *how competitive with Deep Neural Networks can we be with a no learning baseline*?

**Exploring Non-Deep Learning Methods:** In an era dominated by Deep Learning research, I have deliberately pursued work that either avoids or minimizes the use of Deep Learning techniques, as exemplified by [THIRY et al., 2021]. One such study employs random patch-based architectures for classifying images in the ImageNet dataset. This approach removes the need for a Deep Neural Network, by encoding each image patch using a random dictionary and employing a linear classifier for the final classification. The dictionary itself consists of random patches, and the encoding relies on a straightforward hard nearest neighbor assignment. Remarkably, this elementary method achieves a 45% top-1 accuracy on ImageNet, which stands in contrast to the 60% typically achieved by more complex approaches. Note that this approach is competitive with results *prior the Deep Learning era*. This is a quite intriguing result.

Notably, this technique not only outperforms Deep Neural Networks but also bests Scattering Networks [Oyallon, 2016]. This phenomenon calls into question the conventional wisdom surrounding the choice of appropriate invariance modeling through theoretical considerations versus empirical data-driven improvements.

In collaboration with Dr. Gaël Varoquaux, I have also explored the analysis of tabular data [Grinsztajn et al., 2022]. Our published study rigorously compares Random Forests and Deep Learning across standard benchmarks. This research serves to elucidate the distinct assumptions that each algorithm makes about the data, thereby providing insights into conditions that yield similar or disparate performance outcomes, valuable for both Machine Learning practitioners and researchers.

**Neural Network Analysis and Interpretability:** In my pursuit to understand the foundational principles behind the success of Deep Learning, I have explored and proposed a significant number of architectures through various paradigms and concepts. My research in this area commenced during my Ph.D. with Oyallon [2017b], where I conducted empirical analyses of Convolutional Neural Networks to understand the types of invariance implemented in their inner layers. These findings align with existing works [Szegedy et al., 2014, Zeiler and Fergus, 2014]. In Jacobsen et al. [2018], we empirically demonstrated that the Information Bottleneck, as introduced by Tishby et al. [2000], is not necessarily present in state-of-the-art trained Deep Neural Networks. We achieved this by designing the first class of invertible supervised Deep Neural Networks for classification. While this contribution involved only a minor architectural modification, its implications are substantial. Additionally, we examined the

progressive linear separability of classes across the network's depth, finding that it exhibited ongoing improvement, in contrast to mutual information metrics. In collaboration with Dr. Francis Bach and Dr. Lénaïc Chizat, I investigated the empirical effectiveness of lazy training [Chizat et al., 2019]. The concept suggests that under certain regimes, Neural Networks behave as if they are linearized around their initialization, leading to suboptimal performance. Previous work by these co-authors showed that this regime can be circumvented with appropriate initialization, thus enabling "feature learning." I have synthesized several theoretical ideas and proposals based on these concepts in my online lectures [Oyallon, 2021].

**Federated Learning:** In this paradigm, heterogeneous workers transmit their gradients to a central worker for aggregation. The primary challenges involve ensuring privacy and minimizing communication overhead. Recognizing the field's significance, I was introduced to Federated Learning by Eugene, with a specific emphasis on communication reduction strategies. In Tenison et al. [2022], we demonstrated that proper initialization of the head of a Convolutional Neural Network could significantly reduce the need for gradient communication. Additionally, in Legate et al. [2023], we proposed a non-linear aggregation method based on weight masking, which proved beneficial in heterogeneous environments.

This diverse body of work has enabled me to gain knowledge in Signal Processing, Optimization, PyTorch, and Machine Learning, providing me with original and necessary tools to address the challenges of distributed training.

### 1.2.2 Contributions of this manuscript

I will now list the contributions I will discuss in this manuscript. The Chapter 2 proposes a brief introduction to the field of distributed training. The three other chapters involve graphs and/or distributed aspects of training a Neural Network.

**Manifold Learning:** Owing to the generality of the "manifold hypothesis," manifold models offer promising initial solutions for a broad range of problems, from predicting chemical potentials [Atz et al., 2021] to image analysis [Hammond et al., 2011]. The majority of my findings in this field have been counterintuitive (to me, at least), suggesting that geometric Deep Learning requires a robust, principled framework—likely exceeding the scope of simplistic harmonic analysis, as I initially believed. For example, in Oyallon [2020], I demonstrated that learning analytical filters (e.g., filters designed to smooth the signal's envelope) could produce transformations analogous to Fourier Transforms. However, subsequent experiments indicated that these ideas were highly specific to graph settings, as elaborated in a similar paper I later found Venkitaraman et al. [2019]. In Sergeant-Perthuis et al. [2022], we showed that the only non-linear operators

for vector fields that commute with diffeomorphisms are scalar operators. This stands in contrast to signals defined over a manifold, where the corresponding operator can be point-wise non-linear—a phenomenon well-understood in the context of Euclidean translations. Despite these challenging findings, I chose to include this line of research in my manuscript. It provides the rationale for my transition to distributed training and continues to offer valuable insights into distributed learning and graph theory. Chapter 3 will delve into these developments.

**Local Learning:** The concept underpinning Local Learning is the training of Neural Network layers using solely local feedback, eschewing the need for ad-hoc global variables that depend on a global computational step. Conventional back-propagation, though an undebated and well-established baseline, poses challenges in parallelization due to its intrinsically sequential nature. Since 2019, a primary objective of my research has been to devise high-performance alternatives to back-propagation. In Belilovsky et al. [2019], we reexamined a prevalent algorithm, Greedy Learning. Despite its simplicity, the scalability of greedy strategies for high-performance tasks, particularly in vision datasets, remained uncertain until our investigation. This foundational work inspired us to create the Decoupled Greedy Learning [Belilovsky et al., 2020] algorithm—a parallelized version of Greedy Learning designed for module-based training. Our approach has demonstrated competitive performance and computational benefits. However, in scenarios requiring extensive parallelization, we have observed performance degradation. To mitigate this decline, we introduced a mechanism based on Forward Gradient in Fournier et al. [2023] to align the local training dynamics more closely with end-to-end training dynamics. Chapter 4 will elaborate on the various facets of Greedy Learning.

**Decoupled Asynchronous Decentralized Algorithms:** My ANR JCJC Adonis grant has enabled me to initiate a promising line of research, focusing on the integration of asynchrony and decentralized learning, in particular for the training of Deep Neural Networks. This research focus falls under a class of distributed algorithms designed to independently process communications and gradient computations on each worker node. The ultimate aim is to develop efficient implementations capable of overcoming limitations inherent to current hardware, which is by design highly synchronous and, in some senses, constrained to a local maximum in term of computational possibility. In Nabli and Oyallon [2023b], we introduced DADAO, an algorithm for convex objectives that leverages asynchronous updates and communications. This algorithm represents a fruitful blend of the continuized framework [Even et al., 2021] and concepts from ADOM+ [Kovalev et al., 2021b]. While its convergence rates surpass those of concurrent methods, questions about its optimal performance remain. Subsequently, we explored approaches to accelerate the convergence of asynchronous gossip algorithms in Nabli and Oyallon [2023a] by optimizing communication rates based on network-specific constraints. We also developed a

relaxation technique to further enhance this optimization. In Nabli et al. [2023], we proposed an acceleration method for peer-to-peer asynchronous communications, specifically in the context of Deep Neural Network training, which led to performance improvements. This work demonstrates empirically that asynchronous communications can indeed be expedited, even in the domain of Deep Learning. Unlike other methods such as those in Hsieh et al. [2023], which only utilized peer-to-peer communications, our approach incorporates an additional layer of acceleration that has proven to be substantially beneficial. Chapter 5 will delve deeper into these contributions.

## 1.3 Academic service

**Supervision.** Since 2020, I had the chance to be surrounded by excellent students, and I list now those with whom I had a supervision leading to publications:

- Louis Fournier, who is co-supervised with Pr. Sylvain Lamprier and is working on the specific topic of Local Learning via Local losses,
- Léo Grinsztajn, who is co-supervised with Dr. Gaël Varoquaux and tackles the problem of learning from tabular data and in non-differentiable settings,
- Jakob Maier, a master student who had allows me to quickly explore several areas related to Geometric Deep Learning, and even obtained a NeurIPS paper!
- Adel Nabli, who is co-suprevised with Pr. Eugene Belilovsky and works mainly on asynchronous decentralized learning,
- Dr. Stéphane Rivaud, who is my first postdoctoral research, working on scaling-up alternatives to back-propagation, with one of the ambitions to provide fast implementations for distributed training.

**International collaborations.** Through the last 10 years of research, I had the privilege to meet exceptional researchers with whom I built strong international collaborations, and the most noticable collaborations are:

- Dr. Michael Eickenberg mainly on the topic of Greedy Learning and,
- Pr. Eugene Belilovsky on the topic of Greedy Learning and the Scattering Transform.

**Grants.** I had the chance to obtain several grants which allowed me to hire most of my students, and in particular, to be:

- PI of an ANR JCJC (2022-2025) and Emergence SU "ADONIS" (2021-2023), which allowed me to hire for 2 years a post-doctorate researcher and one PhD student.

- Collaborator of VHS (2021-2025), which is an interdisciplinary project involving in particular Dr. Mathieu Aubry, for which the goal is to use AI on historical data
- Partner of SHARP (2023-2027), which is a PEPR (consortium) between multiple university and which will lead to new results on frugal learning.

**Teaching.** Since the end of my PhD, I had the opportunity to teach a couple of lectures about Deep Learning, the most notable being:

- Deep Learning in practice, with Dr. Guillaume Charpiat at the "Mathématiques, Vision, Apprentissage" master (master MVA) in 2019-2020, at Ecole Normale Supérieure de Cachan.
- Advanced topics in Deep Learning in 2020-2023, at Institut Polytechnique de Paris, for the master of Data Science (master M2S) hosted by Ecole Polytechnique.

I have been recruited as a Lecturer at Ecole Polytechnique from 2020 to 2023, where I was mainly a TA: I taught Deep Learning and Machine Learning classes mainly. At Sorbonne University, from 2021 to 2022, I have been teaching the Advanced Machine Learning & Deep Learning class. From 2018 to 2019, I was also an Assistant Professor at CentraleSupélec, where I also taught a Reinforcement Learning class. I've also organized some corporate lectures with Sébastien Loustau right after my Ph.D about Deep Learning and one tutorial about Deep Learning with the Société Française de Statistique.

# Chapter 2

# A short discussion on Distributed Deep Learning

The focus of this manuscript will be devoted on distributing training method for Deep Learning, and thus, I believe a bit more of context is required to understand in more depth the Chapter 4 and Chapter 5.

Within the realm of High Computing Performance environments and distributed algorithms, performance assessment and enhancement are deeply intertwined with hardware, implementation, and algorithmic aspects. These components are closely interconnected, and for instance, a significant reason behind the success of Deep Learning lies in the effective interplay of these elements. This motivates the next discussions in next following three sections.

## 2.1 Hardware Ontology

Understanding the variety of available hardware types for training Deep Neural Networks enables us to identify the most suitable computational resources for specific algorithms or potential of improvements [Bertsekas and Tsitsiklis, 2015]. This understanding is critical when considering the hardware's local memory, global memory, number of cores, and the connectivity between those cores or other components, as this might introduce significant computational overhead or performance drop [LeCun, 2019]. Although certain algorithms are best suited for specific hardware types [Bertsekas and Tsitsiklis, 2015], there are also algorithms for which the ideal hardware has not yet been developed, and in particular those involving distributing asynchronously Neural Networks layers [Lacey et al., 2016]. This section offers a brief ontology of hardware that is by no means exhaustive but aims to provoke thoughtful questions about algorithm design. One take-home message is that most of the available hardware is highly synchronous and aims at behaving like a single machine [Choquette, 2023, 2022].

**Multi-core CPUs.** From a general perspectives, central Processing Units (CPUs) serve as fundamental computational units characterized by few cores, and rapid access to a large, global, and shared memory pool. Consequently, CPUs are most compatible with sequential algorithms which demand fast memory access, as for instance, given by a MapReduce procedure Dean and Ghemawat [2008]. Until the "GPUs hack" [Krizhevsky, 2014] which allowed to unlock GPUs strength, hundreds of machines were combined together as in the seminal paper Le [2013].

**GPUs.** Before the 2010s, Graphics Processing Units (GPUs) were primarily engineered by NVIDIA for computer graphics applications, excelling in scenarios where tasks could be executed in parallel without the need for shared local memory [Krizhevsky, 2014]. Matrix multiplication, a fundamental operation in Machine Learning, is a classic example of an "embarrassingly parallel" problem [Fung and Mann, 2005]. In recent years, connectivity bottlenecks between CPUs and GPUs have been alleviated through hardware and software advancements, creating improved computational environment for a wider range of GPUs friendly algorithms [Mittal and Vetter, 2015].

**Multi-GPUs - Multi-nodes with Infiniband Setting.** The notion of "infinite bandwidth" emerged as a theoretical ideal in distributed systems. Although actual bandwidth is always finite, advancements in interconnect technologies have significantly expanded data transfer rates, effectively allowing for near-instantaneous data exchange between GPUs in a cluster. NVIDIA's NVLink, for instance, can provide a bandwidth of up to 300 GB/s, radically exceeding the capabilities of older PCI Express interfaces [Li et al., 2019a].

This paradigm shift towards an "infinite bandwidth" setting has been a cornerstone in the evolution of scalable Deep Learning algorithms and has dramatically influenced the architecture and feasibility of modern Deep Learning systems: in current clusters, it is standard to have an InfiniBand hardware. While continuously puhsing limitical physics of such clusters, they constrain a lot its architecture.

**TPUs (and NPUs, FGPAs...).** Tensor Processing Units (TPUs) are designed specifically for Google Deep Learning training tasks. TPUs excel at handling the matrix operations commonly found in Neural Network training, making them an attractive choice for specific Machine Learning algorithms: they almost allow a linear scaling in the number of machines involved [Wang et al., 2019], pushing further the limits of GPUs. There are numerous alternatives to TPUs and Deep Learning dedicated hardware, which all aim at proposing extremely optimized low-level routines.

**IPUs.** Intelligent Processing Units (IPUs) are, to my knowledge, one of the most different type of dedicated hardware to Deep Learning and developed by Graphcore. In theory, they allow more parallelization between kernels [Louw

and McIntosh-Smith, 2021] and seem to be a more plausible hardware for asynchronous computations combining model parallelism. However, currently, such hardware would benefit from more training algorithms that would maximize its utility while obtaining high-performance.

**And beyond...** Emerging technologies like quantum computing represent the frontier of computational hardware. While they are not yet fully practical for most tasks, their potential for solving specific types of problems—like optimization seems huge [Humble et al., 2021].

## 2.2 Contrasts of Parallelization Principles

Despite the dominance of synchronous-parallel-computing hardware, I see an opportunity to develop novel, more scalable algorithms. I will proceed to contrast various parallelization principles. At times, the nature of an algorithm may be ambiguous, making it easier to understand one concept in juxtaposition with another. Indeed, it is noteworthy that most algorithms can be articulated in one style or another. For example, a parallel algorithm can be adapted to a distributed setting; however, it may not efficiently and fully utilize this paradigm.

**Model VS Data Parallelism.** Model parallelism and data parallelism are two distinct approaches to parallelizing and implementing algorithms. In model parallelism, a large model is divided into smaller components, and each component is processed independently by different workers. For instance, pipelining [Allan et al., 1995, Huang et al., 2019] is an example of such instance which allows to compute a batch of data per slice of the model, in parallel on potentially multiple GPUs or nodes and which can result in a faster implementation with memory saving. On the other hand, data parallelism involves distributing copies of the model to multiple workers, each processing different subsets of the data, which is aligned GPU computations. While those two approaches focus on different aspect of Neural Networks training, they can also be potentially combined together.

**Parallel VS Distributed Computing.** (see Chapter 1 of Bertsekas and Tsitsiklis [2015]) Parallel computing refers to the simultaneous execution of tasks using multiple processors to solve a single problem efficiently, and corresponds typically to a situation with highly well-connected GPUs. Distributed computing, on the other hand, involves coordinating multiple machines to work on related tasks, often tackling larger problems by dividing them into smaller subproblems (e.g., mapreduce [Dean and Ghemawat, 2008]). While parallel computing focuses on speeding up a single task, distributed computing emphasizes solving bigger problems by harnessing the collective power of interconnected machines. As discussed above, the parallel computing paradigm has

often influenced the design of architectures and computing clusters for Deep Neural Networks.

**Synchronous VS Asynchronous Algorithms.** Synchronous algorithms ensure that all participating processes are synchronized at predefined points during computation. This synchronization can lead to efficient coordination, useful to guarantee a result, but may also introduce bottlenecks if some processes are slower than others. Asynchronous algorithms, on the contrary, allow processes to operate independently without strict synchronization requirements. While this can increase overall throughput (potentially with units like IPUs), it also results in challenges related to consistency and convergence [Dean et al., 2012].

**Deterministic VS Randomized Algorithms.** Closely related to the notion of asynchronous algorithms, the distinction between deterministic and stochastic algorithm is important as the latter provides potential important benefits. Deterministic algorithms produce the same output for a given input and execution conditions every time they are run. In contrast, randomized algorithms benefit from the uncertainty of randomness. While deterministic algorithms guarantee reproducibility, randomized algorithms can often provide faster and more efficient solutions, both from a theoretical and practical point of view [Mishchenko et al., 2022, Defazio et al., 2014, Leblond et al., 2017].

**Centralized VS Decentralized Algorithms.** Centralized algorithms rely on a central worker to make decisions and coordinate processes of plethora of workers [Li et al., 2020]. For instance, a synchronous algorithm is *necessarily centralized*, as a global orchestrator guarantees that each instruction stops at a given step. On the contrary, decentralized environments only rely on local information available on a given node [Nedic and Ozdaglar, 2009].

**Convex VS Deep Models.** Convex objectives are much better understood than their deep counter part: typically, the convexity allows to derive guarantees for the convergence of the algorithm [Nesterov, 2003]. On the contrary, it is pretty difficult to obtain a convergence result for a Deep Neural Network algorithm, whose analysis in general is bounded to obtain a lower-bound via a local analysis of the norm of the gradient $\|\nabla f(x)\|$ of an objective $f$ with parameter $x$ [Bousquet et al., 2004].

**A word on 3D/4D Parallelism.** Emerging paradigms such as 3D or even 4D parallelism [Lai et al., 2023, Li et al., 2021] offer possibilities for enhancing training speed by fusing various levels of low-level parallelization. While these approaches hold significant promise and constitute an important research direction, they do not affect the training dynamic: they consist in improved implementation to compute gradients; by contrast, I would like to propose novel training dynamics with computational advantage, and which could be potentially combined with those approaches.

## 2.3 A brief discussion on decentralized learning algorithms.

Now, that I clarified several concepts I will use widely through this manuscript, I will simply propose a short description of decentralized algorithms (not necessarily asynchronous), starting first with dual ascent, to finally conclude with gradient-based methods, more amenable to Deep Learning.

**Problem setting.** In this manuscript, we study problems of the type

$$\inf_{x\in\mathbb{R}^d} \sum_{i=1}^{n} f_i(x)\,, \tag{2.1}$$

where each $f_i : \mathbb{R}^d \to \mathbb{R}$ is a function with some regularity, e.g., $L$-smooth. Typically, in those finite sum structure Machine Learning problems, each $f_i$ can be written for some labeled data $(\theta_i, y_i)$

$$f_i(x) \triangleq \ell(f(x;\theta_i), y_i)\,, \tag{2.2}$$

where $f$ is some Machine Learning models, e.g., a Deep Neural Network or a linear layer. If $f_i$ is convex, incorporating a $\ell^2$ regularization to Eq. (2.1) given by

$$\inf_{x\in\mathbb{R}^d} \frac{n}{2}\mu\|x\|^2 + \sum_{i=1}^{n} f_i(x)\,, \tag{2.3}$$

could also be obtained by assuming that each $f_i$ is at least $\mu$-strongly convex.

**Connectivity.** In a typical decentralized context, one has a network of $n$ nodes, whose topology is determined by a (potentially directed) graph with edges given by $\mathcal{E} \subset \{1,...,n\} \times \{1,...,n\}$. In this case, one often introduces the concept of connectivity matrice $[\lambda_{ij}]_{ij}$, where $\lambda$ is a $n \times n$ matrix so that $\lambda_{ij} = 0$ iff $(ij) \notin \mathcal{E}$.

From the weights $\lambda_{ij}$, we introduce the corresponding symmetric Laplacian given by

$$\Lambda = \sum_{(ij)\in\mathcal{E}} \lambda_{ij}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T}\,. \tag{2.4}$$

This quantity is key to model connectivity, and will be discussed in more depth in Chapter 5.

**Dual decomposition.** A decentralized problem has typically a separable expression, with a linear constraint (see Bertsekas and Tsitsiklis [2015, Section 3.4]), and one typically considers a problem of the type

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to } & \Lambda x = 0, \end{aligned} \tag{2.5}$$

whose Lagrangian is given by, writing $\Lambda = [\Lambda_1, ..., \Lambda_n]$ by (see Boyd et al. [2011, Chapter 2.2])

$$L(x, y) = \sum_{i=1}^{n} L_i(x_i, y) = \sum_{i=1}^{n} f_i(x) + y^{\mathsf{T}} \Lambda_i x_i \,. \tag{2.6}$$

Then an algorithm would be naturally for some $\alpha^k$

$$x_i^{t+1} = \arg\min_x L_i(x_i, y) \tag{2.7}$$

$$y^{k+1} = y^k + \alpha^k \Lambda x^{k+1} \,. \tag{2.8}$$

If $\Lambda$ represents the constraint of a decentralized problem, then this aims at solving a distributed optimization task with communication constraints. Agarwal et al. [2010] is one of the first works to relate the spectral gap to the convergence speed, when Scaman et al. [2017] derives optimal, synchronous, dual algorithm which reaches optimal rates, in particular thanks to Chebychev acceleration [Saad, 1984] and the standard Nesterov quadratic for deterministic lower-bounds [Nesterov, 2003]. The key is to observe that

$$\inf_{\Lambda x = 0} \sum_{i=1}^{n} f_i(x_i) \iff \sup_y - \sum_{i=1}^{n} f_i^*(e_i^{\mathrm{T}} y \sqrt{\Lambda}) \,. \tag{2.9}$$

Unfortunately, such approaches (e.g., relying on problem structure) are generally not primal gradient based, which makes them difficult to transpose to a Deep Learning setting. Alternating Direction Method of Multipliers (ADMM, introduced in Boyd et al. [2011]) represents a widely used framework aimed at splitting a problem into subproblems via an augmented Lagrangian. While it relies on a proximal step, the idea of using auxiliary variables has application beyond, e.g. Choromanska et al. [2019], Taylor et al. [2016], Zhang et al. [2016].

**Primal methods.** Two seminal works can be considered to distribute computations: on one side Tsitsiklis et al. [1986] which considers

$$\inf_{x_1, ..., x_n} f(x_1, ..., x_n) \,,$$

so that each worker has its own $x_i$ and access to $f$. On the other side, Nedic and Ozdaglar [2009] is one of the seminal works which introduce an optimization procedure which alternates between gradients computations and communications to solve

$$\inf_{\Lambda x = 0} \sum_i f_i(x_i) \,.$$

Koloskova et al. [2020] is a tight analysis of such algorithms in the stochastic heterogeneous setting. Methods like DiGing [Li et al., 2019b], EXTRA [Shi et al., 2015] are examples of algorithms leading to linear rates. A technique to synchronize gradient average is Gradient Tracking [Song et al., 2023], studied

also in the stochastic setting [Koloskova et al., 2021]. Each of these methods can be extended to Deep Learning settings [Lian et al., 2017], yet those methods are highly sequential and synchronous: in this manuscript, we will try to add an additional layer of parallelization, either by parallelizing the algorithm to compute $\nabla f_i(x)$, either by deriving more asynchronous algorithms.

# Chapter 3

# Exploring Geometric Deep Learning: A Harmonic Analysis Perspective

This chapter draws upon the following key publications or preprints:

- Edouard Oyallon. Interferometric graph transform: a deep unsupervised graph representation. In *International Conference on Machine Learning*, pages 7434–7444. PMLR, 2020, whose main idea is presented in Section 3.1
- Grégoire Sergeant-Perthuis, Jakob Maier, Joan Bruna, and Edouard Oyallon. On non-linear operators for geometric deep learning. *Advances in Neural Information Processing Systems*, 35:10984–10995, 2022, whose results are explained in Section 3.2,
- Nathan Grinsztajn, Louis Leconte, Philippe Preux, and Edouard Oyallon. Interferometric graph transform for community labeling. *arXiv preprint arXiv:2106.05875*, 2021a and,
- Nathan Grinsztajn, Philippe Preux, and Edouard Oyallon. Low-rank projections of gcns laplacian. *ICLR Workshop GTRL*, 2021b which are two preprints discussed in Section 3.3.

The main students and collaborators involved are Pr. Grégoire Sergeant-Perthuis, Jakob Maier, Nathan Grinsztajn and Pr. Joan Bruna.

Manifold learning provides an intriguing framework for capturing intricate data structures, particularly those with weak inherent regularity [Shuman et al., 2013, Coifman et al., 2005]. One salient aspect of manifolds $\Omega \subset \mathbb{R}^d$ is their frequent representation as sparse graphs, encapsulating local interactions between $n$ points on the manifold, via a $n \times n$ matrix. Another representation consists in explicitly encoding the action of a group of symmetries reflected by

the manifold data [Satorras et al., 2021]. For both cases, such representations are not arbitrary; they often lead to a smoother representation with respect to the (local) Euclidean metric.

The scientific community has made concerted efforts toward developing a comprehensive mathematical framework for manifold learning on graphs, and in particular to transpose Deep Learning results on graphs [Bronstein et al., 2017, 2021]. A key objective is to develop models with inductive biases leading to favorable properties; this construction is very often done by analogy with the Convolutional Neural Networks class of models. In doing so, three primary approaches have gained prominence:

- **Direct Encoding of the Underlying Geometric Structure** : This approach is predicated on obtaining an explicit mathematical model that captures the group of variabilities [Cohen and Welling, 2016] (or symetries) inherent in the manifold structure. Subsequently, an objective is to design architectures that are inherently invariant to these variabilities. For instance, Convolutional Networks specifically tailored to operate on spherical geometries have been proposed [Cohen et al., 2018]. Further advancements in this direction include architectures that work along parallel transport, thereby encapsulating complex geometric transformations [Bodnar et al., 2022].

- **Leveraging Local Euclidean Norms**: Here, the aim is to encode the manifold's intrinsic geometric structure into an adjacency matrix [Saul and Roweis, 2000], thereby facilitating the application of Graph Convolutional Neural Networks [Kipf and Welling, 2016]. This approach effectively amalgamates local information based on the Euclidean norms, enabling the Neural Network to adapt to the manifold's topology.

- **Utilizing the Spectrum of the Laplacian Matrix** [Hammond et al., 2011]: In this paradigm, the focus shifts to spectral methods, particularly those involving the Laplacian matrix. One commonly employed strategy is to construct filters as sparse linear combinations of Laplacian eigenvectors. This approach has proven efficacious in a range of applications, as demonstrated in recent studies [Wang and Zhang, 2022].

**Discussion and Organization of this Chapter:** In this chapter, I discuss the outcomes of my efforts to apply principles from harmonic analysis to manifold learning (see Section 3.1), and in particular, to develop architectures and principles simpler for learning on manifold structures. The Section 3.3 will briefly mention some attempt to extend those ideas to generic problems on graphs. In particular, we obtain a negative result to construct generic non-linearity in Section 3.2. While this research spans multiple years, a pivotal juncture was reached during the supervision of a Master's student, Jakob Maier. It was through this collaboration that several significant milestones in this line of inquiry were achieved.

## 3.1 Interferometric Graph Transform: a Deep Unsupervised Graph Representation

In Oyallon [2020], I proposed to study a simple, straightforward question: can we generalize Fourier Transform to manifolds? Defining a notion of Fourier Transform in manifolds is a difficult question because researchers have tried to obtain from the Eigen-vectors of the Laplacian. However, by design, standard spectral methods suffer from several inherent issues, which also apply to the Euclidean domain. A first issue is the lack of topology of the Laplacian's eigenvectors. For the sake of illustration, observe that for a smooth $f \in L^2(\mathbb{R}^k), k \geq 0$, the Fourier transform of its Laplacian satisfies

$$\forall \omega \in \mathbb{R}^k, \widehat{\Delta f}(\omega) = -\|\omega\|^2 \hat{f}(\omega) \, .$$

Here, the topology of the eigenbasis (e.g., a cosine family) is difficult to exhibit from its corresponding eigenvalues. For instance, two rather different frequencies (e.g., $\omega_1 \neq \omega_2$) with the same amplitude (e.g., $\|\omega_1\| = \|\omega_2\|$) will not be distinguished by a spectral clustering algorithm based solely on $\|\omega\|$. This typically leads to filters which are isotropic and not selective to a specific direction, which also holds for spatial methods [Bronstein et al., 2017]. A second issue is that the graph convolution employs filters which are built from local operators such as a Laplacian matrix: this typically leads to a smoothing operator [Kampffmeyer et al., 2019, Li et al., 2018, NT and Maehara, 2020, Wu et al., 2019]. Thus, in those settings, spectral GCNN lose the ability to discriminate high-frequency attributes of a signal, which are also usually unstable and thus difficult to capture [Mallat, 1999]. In our work, we address those two issues by learning a complex-valued isometry in the spectral domain, which has, for instance, the ability to recover the spectral topology of 2D frequencies, without incorporating any specific prior: the filters are anisotropic and smooth in frequencies. The main idea is to form some pairs of conjugated filters, as a cosine eigen-vectors can be deduced from a sine eigen-vectors via a Hilbert Transform [Cizek, 1970].

**A criterion based on the eigen-vectors smoothness.** In Oyallon [2020], I proposed a criterion to pair real-valued basis of eigen-vectors, in order to further propose an ordering of the eigen-vectors. To do so, for a permutation $\pi$ of $\{1, ..., 2d\}$, we introduce the pairing cost

$$\begin{aligned} C(\pi) &= \sum_{i=1}^{d} \sum_{k=1}^{2d+1} \sqrt{e_{\pi[2i]}[k]^2 + e_{\pi[2i-1]}[k]^2} \\ &= \sum_{i=1}^{d} \|e_{\pi[2i]} + \mathbf{j} e_{\pi[2i-1]}\|_1 \, . \end{aligned} \tag{3.1}$$

We then consider the matrix $\mathcal{F} = \{\mathcal{F}_i\}_{i \leq 2d+1}$ whose columns are defined by

$\forall 1 \leq i \leq d$,

$$\begin{cases} \mathcal{F}_i & = e_{\pi^*[2i]} + \mathbf{j} e_{\pi^*[2i-1]} \,, \\ \mathcal{F}_{2d+1-i} & = e_{\pi^*[2i]} - \mathbf{j} e_{\pi^*[2i-1]} \,, \\ \mathcal{F}_{2d+1} & = e_{2d+1} \,. \end{cases} \tag{3.2}$$

Observe that if $i \leq d$, then $\overline{\mathcal{F}_i} = \mathcal{F}_{2d+1-i} \in \mathbb{C}^{2d+1}$. It is clear that the matrix $\mathcal{F}$ is unitary on $\mathbb{C}^{2d+1}$. For illustration purpose, consider the graph $\mathcal{G}$ of a grid of length $2d+1$ with periodic boundary condition, an eigenbasis of its discrete Laplacian is clasically given, for $k \leq d, m \leq 2d+1$, by

$$e_{2k-1}[m] = \sqrt{\frac{1}{2d+1}} \cos(\frac{\pi}{2d+1}(m - \frac{1}{2})2k) \,, \tag{3.3}$$

$$e_{2k}[m] = \sqrt{\frac{1}{2d+1}} \sin(\frac{\pi}{2d+1}(m - \frac{1}{2})2k) \,, \tag{3.4}$$

$$e_{2d+1}[m] = \frac{1}{\sqrt{2d+1}} \,. \tag{3.5}$$

In this case, we know that an optimal permutation $\pi^*$ is given by $\pi^*[n] = n$, which leads to pair of eigen-vectors: $\mathcal{F}_i[m] = \sqrt{\frac{1}{2d+1}} e^{\mathbf{j} \frac{\pi}{2d+1}(m-\frac{1}{2})2i}, i \leq d$.

This thus justifies the terminology Fourier Transform for $\mathcal{F}$ (up to a phase multiplication) as one can recover the Discrete Fourier Transform: our method has a natural interpretation in the Euclidean case. Pairing those eigen-vectors allows to introduce an asymetry between the real and imaginary part of our spectral operator, which will be useful and necessary for learning a complex unitary operator.

**Limitations:** My work Oyallon [2020] has been applied successfully on simplistic task by learning a very similar transform to a Scattering Transform [Oyallon et al., 2018]. Unfortunately, I found out with Jakob Maier that this type of strategy was numerically unable to recover any interesting pairing beyond the case of $\mathbb{R}^d$: the case of the spherical harmonics of $\mathcal{S}^{d-1}$ totally fails to pair the real and complex parts of the spherical harmonics. Furthermore, it would be dependent on the coordinate system. This limits a lot this method, which I initially believed to be principled.

## 3.2 On Non-Linear operators for Geometric Deep Learning

Convolutional Neural Networks are particularly effective due to the inherent benefits of convolutional operators [Mallat, 2016]. Firstly, these operators naturally align with the symmetries present in the data, which allows to build relevant invariants. Secondly, they introduce useful inductive biases that enable efficient computations and reduce the atmount of parameters to learn: it allows to reduce the sample complexity of the learning task.

In the case of data defined on manifolds, these symmetries are not as straightforward. There are, however, two types of objects that can have meaningful actions with objects defined manifold $\Omega$. These objects are based on diffeomorphisms, denoted by $\phi \in \text{Diff}(\mathcal{M})$.

- On one hand, the action of the diffeomorphism group on signals $x \in L^2(\Omega)$ (meaning that $x(u) \in \mathbb{C}$, for $u \in \Omega$), which can naturally act via

$$L_\phi x(u) \triangleq x(\phi^{-1}(u)) \,. \tag{3.6}$$

- On the other hand, the vector fields $x \in L^2(\Omega, T\Omega)$ (meaning that $x(u) \in T_u\Omega$ for $u \in \Omega$, and $T\Omega$ is the tangent bundle of $\Omega$), whose push-forward action is then given by

$$L_\phi x(u) \triangleq d\phi(u).x(\phi^{-1}(u)) \,. \tag{3.7}$$

We ask the simple question: can we build Neural Networks which are covariant with those actions? For instance, in the case $\Omega = \mathbb{R}^d$, Neural Networks, restricting the group of symmetries to translations, the only affine operators which commute with the action of translations are convolutions. In fact, Bruna [2013] showed that the only operators of $L^2(\mathbb{R}^d)$ which commute with the group of deformations are pointwise non-linearities. Such operators also naturally commute with the translation group, as it is a subgroup of diffeomorphism. This justifies the discussion about Convolutional Neural Networks above.

Getting back to the general setting, in fact, any symmetry group is by construction a subgroup of $\text{Diff}(\Omega)$. However, there is a distinction between groups which are locally compact and groups which are not: for instance, [Yarotsky, 2022] observed that Convolutional Neural Networks are a class of dense operators in the commutant with translations. To be amenable to any neural networks designed on a manifold, a non-linearity will thus have to commute with $\text{Diff}(\Omega)$. In a collaboration with Jakob Maier, Pr. Grégoire Sergeant-Perthuis and Pr. Joan Bruna, we focused on those settings. We extended the results of Pr. Joan Bruna's PhD:

**Theorem 3.2.1** (Scalar case)**.** *Let $\mathcal{M}$ be a connected and orientable manifold of dimension $d \geq 1$. We consider a Lipschitz continuous operator $M : L^p_\omega(\mathcal{M}, \mathbb{R}) \to L^p_\omega(\mathcal{M}, \mathbb{R})$, where $1 \leq p < \infty$. Then,*

$$\forall\, \phi \in \textit{Diff}(\mathcal{M}) : \; ML_\phi = L_\phi M$$

*is equivalent to the existence of a Lipschitz continuous function $\rho : \mathbb{R} \to \mathbb{R}$ that fulfills*

$$M[f](m) = \rho(f(m)) \qquad a.e.$$

*In that case, we have $\rho(0) = 0$ if $\omega(\mathcal{M}) = \infty$.*

This result is consistent with the result of Bruna [2013], and corresponds to a generalization to manifolds. This result does not mean that for any group, pointwise non-linearities are optimal. For instance, Eickenberg et al. [2022] used a

$\ell^2$-pooling to combine spherical harmonics which allows to smooth the enveloppe after a specific class of convolutions. However, the following result, was to me a negative result: it indicates that the only generic operator on vector fields which commute with the action of diffeomorphism is a scalar multiplication. This indicates that for vector field, the non-linearity will have to be selected in a per case basis:

**Theorem 3.2.2** (Vector case)**.** *Let $\mathcal{M}$ be a connected and orientable manifold of dimension $d \geq 1$. We consider a continuous operator $M : L^p_\omega(\mathcal{M}, T\mathcal{M}) \to L^p_\omega(\mathcal{M}, T\mathcal{M})$, where $1 \leq p < \infty$. Then,*

$$\forall\, \phi \in \mathit{Diff}(\mathcal{M}) :\ ML_\phi = L_\phi M$$

*is equivalent to the existence of a scalar $\lambda \in \mathbb{R}$ such that*

$$\forall f \in L^p_\omega(\mathcal{M}, T\mathcal{M}) :\ M[f](m) = \lambda f(m) \quad a.e.$$

**A brief comment on the proof technique:** Probably novel, the main idea of the proof we found was to consider diffeomorphisms with compact support, and to show that a sequence of diffeomorphism can "contract" a neighborhood into a point. This is very similar to a homotopy, where simply connected set can be contracted to a point - except that we had the constraint to have a local support.

**Relation of this result with Geometric Deep Learning.** In this work, we have fully characterized non-linear operators which commute under the action of smooth deformations. In some sense, it settles the intuitive fact that commutation with the whole diffeomorphism group is too strong a property, leading to a small, nearly trivial family of *non-linear* intrinsic operators. While on their own they have limited interest for geometric deep representation learning, finding commutants can 'upgrade' any family of linear operators associated with any group $G \subset \mathrm{Diff}(\mathcal{M})$ into a powerful non-linear class — the so-called GDL Blueprint in Bronstein et al. [2021]. Also, this result is a first step towards characterizing the non-linear operators which commute with Gauge transformations and could give useful insights for specifying novel Gauge invariant architectures.

## 3.3 Follow-up works

Mainly with Nathan Grinsztajn, I started investigating if this research could be also extended to Graph Neural Networks, and in particular to the task of community detection: typical models assume a low-rank structure of the Laplacian using a Stochastic Block model [Abbe, 2017], which similarly to a Laplacian defined over a manifold suggests that the very first eigen-values are important and contain most of the discriminative information. Unfortunately, there is a lack of good numerical benchmarks to really demonstrate an advantage for the method: the standard tasks can very often be solved with simple methods which almost do not use the graph structure.

**Low rank laplacian** In Grinsztajn et al. [2021b], we find that one can achieve comparable performance using only a subset of the graph's spectral frequencies, thereby reducing computational complexity. We make two additional contributions. First, we demonstrate that much of the information leveraged by Graph Convolutional Networks for community detection is concentrated in the earliest eigenvectors of the Laplacian matrix. Second, we find that a simple Multi-Layer Perceptron model, when fed with hand-crafted features representing these low-frequency eigenvalues, is capable of handling transductive datasets effectively. These observations suggest a more efficient, yet equally effective, approach to community detection.

**Low rank community** In Grinsztajn et al. [2021a], we develop a simplified framework for analyzing node labeling tasks. It focuses on a toy example that relies on the rank of the Laplacian matrix to provide both theoretical and empirical validations for our methods. This framework is backed by concentration bounds and is supported by numerical evidence. To showcase the framework's capabilities, we extend the representation method of Oyallon [2020] so that it is suitable for graphs. This new method is effective for community labeling tasks and does not require any prior community labels. Our results show that our method performs better than existing Graph Convolutional Networks (GCNs) in certain settings and is competitive in standard benchmark tests.

# Chapter 4

# Model-parallelism: Local Learning

This chapter is based on the following work:

- Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pages 583–593. PMLR, 2019 which is discussed in Section 4.1,
- Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. In *International Conference on Machine Learning*, pages 736–745. PMLR, 2020,
- Eugene Belilovsky, Louis Leconte, Lucas Caccia, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns for synchronous and asynchronous distributed learning. *arXiv preprint arXiv:2106.06401*, 2021 are both showcased in Section 4.2,
- Louis Fournier, Stéphane Rivaud, Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Can forward gradient match backpropagation? In *Fortieth International Conference on Machine Learning*, 2023 is finally discussed Section 4.3.

The main students and collaborators involved are Louis Fournier, Dr. Stéphane Rivaud, Pr. Eugene Belilovsky and Dr. Michael Eickenberg.

Local Learning is a strategy to propose layer updates only based on local information available on a specific layer (e.g., a single module or a cascade of modules), without the need for a global orchestration which would require a synchronization from upper, and, bottom layers. Typically, Local Learning allows us to assume that a layer is fully stored on a given machine, and has limited feedbacks with the previous and next layers. In the context of distributed training, the ultimate goal of a local learning strategy is to distribute, asynchronously, communication and computations to train a Deep Neural Network.

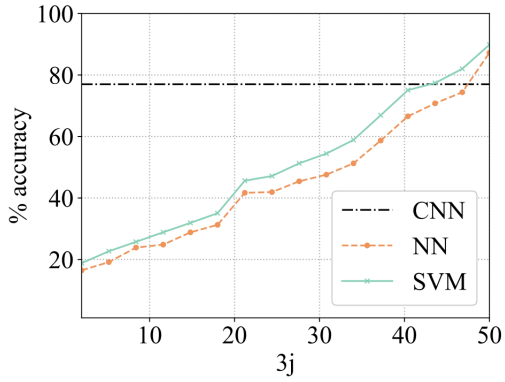


Figure 4.1: Progressive separability as measured by a Nearest Neighbor or a Linear SVM classifier, for a ResNet-152 on the ImageNet dataset, at a depth $3j$. The figure si collected from Jacobsen et al. [2018].

In this chapter, we discuss my contributions to Local Learning and in particular via Greedy Learning, which is the idea of using a greedy, layer-per-layer training strategy.

**Local Learning: A Step Towards Distributed Computing for Deep Neural Network Training** Mini-batch SGD is typically sped-up on GPUs and this procedure is often called "embarassingly-parallel" [Bottou and Bousquet, 2007]. This is innacurate as the back-propagation of each sample still requires a significant amount of memory and by essence, backpropagation is a highly synchronous procedure [LeCun et al., 1989]. In fact, the backpropagation algorithm is well-known for this computational inefficiency [Jaderberg et al., 2017a, 2014], particularly due to its limited ability to parallelize computations between subsequent layers (model parallelism). This makes it challenging to develop a distributed version of backpropagation for training Deep Neural Networks. Additionally, there are theoretical gaps in our understanding of the algorithm's dynamics, even for networks with simply one hidden layer: the objective of the inner layers is unclear, making it even more difficult to design an appropriate optimizer [Szegedy et al., 2014]. In fact, the training dynamic can have complex behavior leading to very different solutions [Godfrey et al., 2022].

**Progressive linear separability** Deep Neural Networks are composed of cascading affine operators and point-wise nonlinearities, which together construct classification invariants. However, the specific process by which these invariants are created remains an open question. Several works [Oyallon, 2017b, Zeiler and Fergus, 2014, Jacobsen et al., 2018] have already observed that Deep Neural Networks designed for image classification naturally establish progressive invariance to various forms of image variability (see Figure 4.1, obtained

from my work [Jacobsen et al., 2018]). This means that the degree of invariance improves as one measures it across the layers of the network. One of the simplest forms of invariance to assess is linear separability [Han et al., 2021]. Specifically, linearization is a commonly observed phenomenon, particularly in supervised Convolutional Neural Networks, for creating the final classification invariant. However, the question of whether this property of linear separability serves reciprocally as a good criterion for state-of-the-art classification performance was left unanswered until the work Belilovsky et al. [2019].

**Discussion and Organization of This Chapter:** The chapter begins with Section 4.1, which explores the concept of Greedy Learning, a method that trains a Deep Neural Network one layer at a time using a specialized auxiliary classifier. Following that, Section 4.2 delves into an extension of Greedy Learning that is geared towards distributed training, referred to as Decoupled Greedy Learning. This approach enables the parallel training of individual layers. Finally, Section 4.3 presents various attempts to improve the accuracy of Decoupled Greedy Learning, using a forward gradient technique.

## 4.1 Greedy Learning

**Algorithm 2** This a simple and concise description of a Greedy Layerwise training of Convolutional Neural Networks, for obtaining the parameters obtained from our optimization procedure $\{(\theta_j^*, \gamma_j^*)\}_{j \leq J}$ over one pass of the data $\mathcal{D}$

**Input:** Training samples $\mathcal{D} = \{(x_0^n, y^n)\}_{n \leq N}$
**Initialize:** Parameters $\{\theta_j, \gamma_j\}_{j \leq J}$.
**for** $j \in 0..J-1$ **do**
  Compute $\{x_j^n\}_{n \leq N}$ (via Equation (4.1))
  $\theta_j^*, \gamma_j^* = \arg\min_{\theta_j, \gamma_j} \sum_n \ell_j(\theta_j, \gamma_j, x^n)$ (via Equation (4.2))
**end for**

**Motivation Rooted in Neural Network Theory:** This research aims to delve into the mechanics of Deep Neural Network optimization. We wanted to challenge the implicit suggestion of Bengio et al. [2006] that greedy training of Deep Neural Networks is infeasible, our team—comprising Pr. Eugene Belilovsky, Dr. Michael Eickenberg, and myself—sought to understand the underlying limitations and potentially propose a remedy. There is a wide line of attempt to get this type of methods to work, as suggested by: [Ivakhnenko and Lapa, 1965, Greff et al., 2016, Huang et al., 2018, Malach and Shalev-Shwartz, 2018, Arora et al., 2014]. Initially, this was a long-term research direction that I had planned to pursue for several years. However, this simple greedy strategy appeared to be surprisingly effective at training state-of-the-art architectures on the large scale ImageNet dataset.

**Re-examining Greedy Layer-wise Training:** In our paper Belilovsky et al. [2019], we took a fresh look at the foundational concept of training a Deep Neural Network layer by layer. The core idea revolves around pre-training each layer with a supervised criterion obtained from labeled data $(x, y)$, defined recursively for parameters $\theta_j$ and layer $f_j$ by

$$x_{j+1} = f_j(x_j, \theta_j)\,, \tag{4.1}$$

and an intermediary loss $\ell_j$ is introduced for this process, as well as an *auxiliary neural network* $g_j$ which is parametrized by $\gamma_j$, so that

$$\ell_j(\theta_j, \gamma_j, x, y) \triangleq \ell(g_j(f_j(x_j, \theta_j), \gamma_j), y)\,. \tag{4.2}$$

Each of these intermediary losses, $\ell_j$, is used to train one block of layers $f_j \circ g_j$ at a time, progressing sequentially to ensure complete convergence. This procedure is detailed in Algorithm 4.1. While the function $f_j$ is dictated by the architecture we're aiming to train (usually representing a single layer or a block of layers), $g_j$ serves as an *auxiliary classifier*, directing the learning signal (here, supervision via labels) during its training.

**Key Components:** Interestingly, our approach enables the training of high-performing Neural Networks for large-scale image classification tasks. Traditionally, such layer-wise methods were mainly used for initializing neural networks and often relied on unsupervised learning. For our experiments, we consider $\{f_j\}_{j \leq J}$ to represent architectures like VGG or ResNet. We find that the choice of auxiliary classifiers, denoted as $\{g_j\}_{j \leq J}$, plays a critical role in the overall performance. Specifically, the design of the final "head" (e.g., the last linear layer and pooling) has a substantial impact. More precisely, we discovered that reducing the spatial windows of the averaging module in these auxiliary networks was key to achieving high performance. This observation aligns well with the understanding that the composed functions $\{g_j \circ f_j\}_{j \leq J}$ lead to neurons with smaller receptive fields. A second key element is the depth of $g_j$, which can significantly boost performance without requiring very wide hidden layers.

**Results** Figure 4.2 presents the results of our Greedy Learning strategy when applied to ImageNet, for which we varied the depth of the corresponding auxiliary classifiers $g_j$. Before training each variants, we fixed the architecture of the $g_j$, only adapting the input size to the output of the layer $f_j$. We report the accuracy for 1, 2 and 3 hidden layers, at various depths. Increasing the depth of the auxiliary leads to good results: as aligned with standard empirical observations, depth matters. There is also a quite different behavior between the one layer auxiliary training and others, which performs significantly worse. We obtain the surprising result that training a sequence of 1-hidden layer Convolutional Neural Networks allows to compete with the AlexNet baseline [Krizhevsky et al., 2012]. The validation accuracy saturates after 6 layers, which is expected.

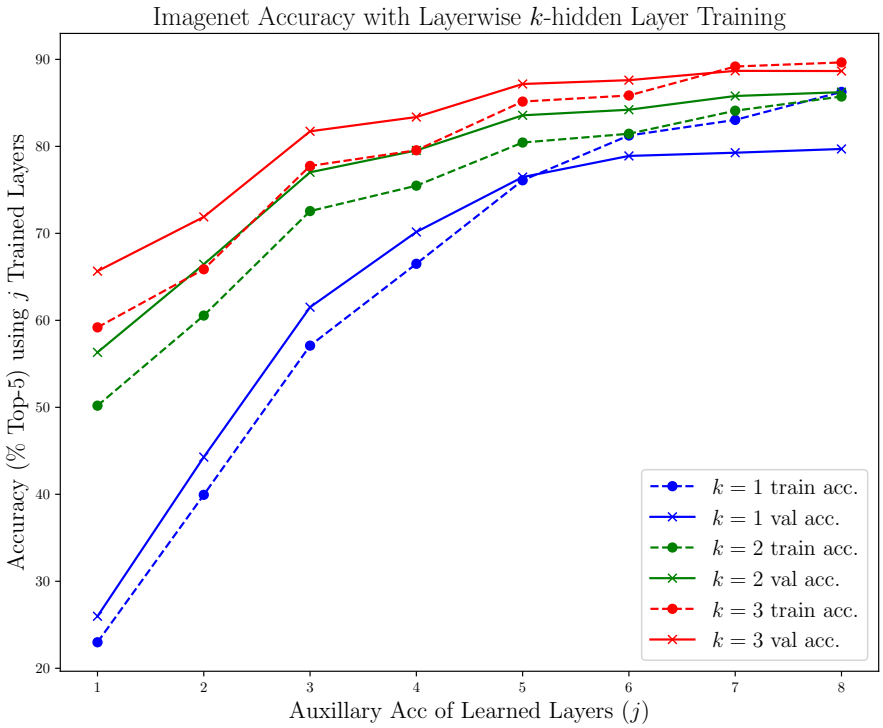


Figure 4.2: Training and validation accuracy of a VGG-like architecture trained on the ImageNet dataset, as classifier by the auxiliary classifier $g_j$.

**Theoretical Significance:** Our research successfully demonstrates that Neural Networks with a single hidden layer can achieve noteworthy performance, which is of interest for current theoretical understandings of Deep Neural Networks. Indeed, existing studies point to an asymptotic regime when the width grows arbitrarily, often referred to as a mean-field regime [Chizat and Bach, 2018], that is closely connected to the problem of training a single-layer Neural Network. These theoretical insights combined with some level of performance guarantees have spurred a substantial body of research [Barak et al., 2022, Ergen and Pilanci, 2021, Malach and Shalev-Shwartz, 2019]. The empirical results of our work serve as a practical validation for this theoretical line of inquiry, emphasizing its importance.

## 4.2 Decoupled Greedy Learning

Via a modification of the previous algorithm, we introduce a parallelization strategy for parallel, layer-wise training as opposed to the conventional sequential approach. Specifically, the model forwards activations through subsequent

**Algorithm 4** This is a synchronous Decoupled Greedy Learning algorithm for updating model parameters $\{(\theta_j, \gamma_j)\}_{j \leq J}$ during one pass over a dataset $\mathcal{D}$.

**Input:** Training samples $\mathcal{D} = \{x_0^n, y^n\}_{n \leq N}$
**Initialize:** Parameters $\{\theta_j, \gamma_j\}_{j \leq J}$.
**for** $(x_0, y) \in \mathcal{D}$ **do**
  **for** $j \in \{1, ..., J\}$ **do**
    $x_{j+1} \leftarrow f_j(x_j, \theta_j)$.
    Compute $\nabla_{(\gamma_j, \theta_j)} \ell_j(\theta_j, \gamma_j, y, x_j)$.
    $(\theta_j, \gamma_j) \leftarrow$ Update parameters $(\theta_j, \gamma_j)$.
  **end for**
**end for**

layers without interruption, enabling the simultaneous training of multiple layers. The efficiency of this parallelized approach is empirically validated in our work Belilovsky et al. [2020], where we report not only improved accuracy on the ImageNet dataset but also an enhancement in computational resource utilization.

| Model (training method) | Top-1 | Top-5 |
|---|---|---|
| VGG-13 ($K = 10$) | 64.4 | 85.8 |
| VGG-13 ($K = 4$) | **67.8** | **88.0** |
| VGG-13 (backprop) | 66.6 | 87.5 |
| VGG-19 ($K = 4$) | 69.2 | 89.0 |
| VGG-19 ($K = 2$) | **70.8** | **90.2** |
| VGG-19 (backprop) | 69.7 | 89.7 |
| ResNet-152 ($K = 2$) | **74.5** | 92.0 |
| ResNet-152 (backprop) | 74.4 | **92.1** |

Table 4.1: Top-1 and Top-5 accuracy on the ImageNet datasets of respectively a VGG-13, a VGG-19 and a ResNet-152 for various split $K$. In bold, we indicated the best performance, and we remind that our models have been trained with a reduced learning rate scheduler, which explain the fluctuations with the numbers reported in the works we reproduced.

**The three locks to remove which limit back-propagation parallelization.** Traditional back-propagation algorithms have several inherent bottlenecks identified by Jaderberg et al. [2017b], making them less suited for parallel and distributed computing environments. First among these is the 'backward lock,' a constraint that prohibits the updating of model weights until a complete propagation of forward and backward activations and gradients has been completed. This lock makes makes difficult model parallelism beyond pipelining [Lai et al., 2023], as weight updates have to wait for the entire cycle to finish. Several concurrent works address this issue with a relative success, see Huo et al. [2018c,a], Choromanska et al. [2019], Nøkland [2016a].

Second, the 'update lock' specifies that a model cannot be updated until the forward signal has passed through all layers to reach the network's output. This restriction limits the adaptability of the model, as it can not incorporate new information until the existing data has been fully processed. Again, this constraint limits the speed and requires an additional synchronization step which is not desirable. At the same time that this work was submitted, several interesting approaches based on delayed gradients have appeared, namely Huo et al. [2018b], Zhuang et al. [2021a,b]

Lastly, the 'forward lock' is less well-defined but constrains how information can propagate forward in the network, creating additional hurdles for efficient computation. The forward lock basically states that a forward pass can only happen in a specific and sequential order. It maintains a synchrony and a need for a global orchestration, as each step of the backpropagation must be processed in a specific order and this is not again desirable.

Removing those three locks have been a key motivation for this research, in the specific setting of supervised Deep Neural Networks.

**Related Research on Feedback Alignment and Direct Feedback Alignment:** Feedback Alignment [Lillicrap et al., 2014b] and Direct Feedback Alignment [Nøkland, 2016b] are alternative training methods that circumvent the need for traditional back-propagation to update neural network weights. Feedback Alignment employs a layer-wise strategy that uses random feedback to approximate the gradients needed for back-propagation. Direct Feedback Alignment, a direct extension of Feedback Alignment, distinguishes itself by sending the output of the final layer directly to each individual preceding layer, instead of operating on a layer-wise basis.

Though these methods successfully overcome the 'backward lock' by enabling weight updates without waiting for a complete forward-backward cycle, they do not alleviate the 'update lock,' which requires the forward signal to reach the final layer before model updates can occur. While both Feedback Alignment and Direct Feedback Alignment have been evaluated in Deep Learning contexts [Launay et al., 2020], the efficacy and potential extensions of these methods on more demanding benchmarks remain ambiguous [Han et al., 2020]. This suggests that while these methods may offer some advantages in specific scenarios, they have limitations that prevent them from fully replacing traditional back-propagation in broader applications. We thus had explorer Decoupled Greedy Learning as a potential solution that can harness the benefits of Feedback Alignment and Direct Feedback Alignment without the associated performance compromises.

**Method: avoiding the reliance on upper modules** A major obstacle to update unlocking is the heavy reliance on the upper modules for feedback. To address this issue, we the joint learning objective of Greedy Learning to not require global feedback: the Algorithm 3 is naturally update and backward unlock. Our greedy learning objective can be then thought as being solved with an alternative optimization algorithm, which permits decoupling the computa-

tions and achieves backward and update unlocking. It can be augmented with replay buffers [Lin, 1992] to permit forward unlocking which is a challenge not effectively addressed by any of the prior work.

**Results of Decoupled Greedy Learning** For our experiments, we utilized a streamlined training scheduler that operated over a condensed 45-epoch training period. Our experimental validation was primarily conducted on the standard ImageNet dataset, and the outcomes are elaborated in Table 4.1. We selected three well-established neural network architectures for this purpose: VGG-13, VGG-19, and ResNet-152. The parameter $K$ signifies the number of subdivisions or 'slices' into which a given Convolutional Neural Network was partitioned. Our results indicate a clear trade-off: increasing the number of splits $K$ generally led to a noticeable decline in the model's accuracy. This effect was particularly pronounced in the case of VGG-13.

**Practical Advantages of Decoupled Greedy Learning** Upon implementing the Decoupled Greedy Learning strategy, we identified two distinct advantages. The first is related to computational distribution: Decoupled Greedy Learning allows for the execution of the algorithm across multiple machines in a model parallelism fashion, each dedicated to the computations related to a specific layer of the neural network. This facilitates a form of horizontal scaling that is particularly beneficial for large-scale applications. The second advantage is more nuanced but equally impactful. Decoupled Greedy Learning enables model parallelism, offering the flexibility to dynamically load and unload network weights. This allows for the training of substantial neural architectures even on hardware with limited resources. For instance, we successfully trained a ResNet-152 model on a single GPU by dividing the model into two halves, achieving this with only a negligible loss in accuracy, as indicated in Table 4.1.

**Significance for Biological Plausibility** Decoupled Greedy Learning offers a solution to the weight transport problem, an issue that has long been a point of contention in the quest for biologically plausible neural networks. This issue is most often cited in critiques of the back-propagation algorithm, which is widely considered biologically implausible due to the weight transport problem and other factors [Bartunov et al., 2018, Nøkland, 2016a, Lillicrap et al., 2014a, Lee et al., 2015, Ororbia et al., 2018, Ororbia and Mali, 2019]. While alternative methods such as Directed Feedback Alignment have been proposed to sidestep this problem, these approaches have yet to demonstrate scalability to large datasets. For example, existing methods have achieved a top-5 accuracy of only 17.5% on ImageNet, in contrast to the reference model's 59.8% accuracy [Bartunov et al., 2018]. Concurrently, there has been work focused on biologically plausible models, such as the studies by Nøkland and Eidnes [2019], which build upon previous results from Mostafa et al. [2018]. Furthermore, note that these models have not explored the capabilities for unlocking or asynchronous training and currently lack the scalability to manage datasets

as large as ImageNet. Therefore, the development and application of Decoupled Greedy Learning present an avenue for achieving biologically plausible, scalable models, filling an interesting gap in the existing literature.

## 4.3 Improving Decoupled Greedy Learning with Forward Gradient

One of the central questions in my research on Decoupled Greedy Learning involves closing the performance gap with traditional End-to-End training. More specifically, I tried to answer to the following question: can DGL be competitive using solely a limited amount of resources? Preliminary experiments indicated that if $g_j$ is designed so that $g_j \circ f_j$ has equivalent depth to the targeted architecture, then it becomes feasible to match the performance of a standard End-to-End setting. This suggests that the *quality of the approximation* of the End-to-End gradient may be a key criterion for local learning to reach End-to-End levels of accuracy. However, this approach is not without its drawbacks, as it incurs considerable computational overhead without a clear justification.

In Fournier et al. [2023], our work started with the simple observation that a well-defined goal of a local method at depth $j$ would be to approximate $\nabla(\ell \circ f_J \circ ... \circ f_{j+1})$ with $\nabla(\ell \circ g_j)$, or in another words, to obtain that

$$\nabla(\ell \circ g_j) \approx \nabla(\ell_J \circ f_J \circ ... \circ f_{j+1})\,. \tag{4.3}$$

However, in a Greedy Learning setting, a layer has a limited access to feedbacks from upper layers, which makes this approximation task almost infeasible. In the following, we will write $F_j \triangleq \ell \circ f_J \circ ... \circ f_{j+1}$.

| Dataset | CIFAR-10 | | ImageNet32 | |
|---|---|---|---|---|
| Space (Method) | Activ. | Weight | Activ. | Weight |
| End-to-End | **94.3 ± 0.1** | | **53.7** | |
| Local, CNN | 79.0 ± 1.0 | 88.0 ± 0.4 | 7.3 | **40.0** |
| Local, MLP | **84.7 ± 0.3** | **88.7 ± 1.2** | **21.8** | 37.4 |
| Local, Linear | 46.7 ± 2.5 | 86.1 ± 1.4 | 10.0 | 23.3 |
| NTK, CNN | 37.7 ± 0.1 | **50.1 ± 1.0** | 2.0 | 3.6 |
| NTK, MLP | **50.3 ± 0.4** | 49.7 ± 0.6 | **7.5** | **3.9** |
| NTK, Linear | 49.9 ± 1.0 | 48.3 ± 0.7 | 4.2 | **3.9** |
| Gaussian | **38.9 ± 0.9** | **50.0 ± 0.8** | **4.9** | **4.9** |
| Rademacher | 38.0 ± 1.5 | 49.8 ± 0.2 | **5.5** | 4.6 |

Table 4.2: Accuracy on CIFAR-10 and ImageNet-32 using a Forward gradient for either the weight update or the activation update. Best performances are highlighted in bold.

**Introduction to the Forward-Gradient Algorithm.** The Forward-Gradient Algorithm [Franceschi et al., 2017] offers a way to compute directional derivatives along a given direction $u$, which is represented as $\langle \nabla f(x), u \rangle u$. While back-propagation is commonly used for this purpose, the Forward-Gradient Algorithm achieves the same result using only forward passes, which involve propagating both activations and directional derivatives. Essentially, the algorithm applies the chain rule to compute $d(f \circ g)(x).u = df(g(x)).(dg(x).u)$ through a straightforward forward computation, which involves first computing $g(x)$ and $dg(x).u$, which are then propagated to the next layer, which is here $f$. This method adds a slight computational overhead due to the propagation of gradients. In Ren et al. [2023], the Forward-gradient algorithm has been proposed as a plausibly biological alternative to back-propagation, which scales on ImageNet, as it leads to an unbiased estimate of the gradient. Indeed, as also noted in Silver et al. [2021], Baydin et al. [2022] if $u \sim \mathcal{N}(0, \mathbf{I}_d)$, it is straightforward to note that

$$\mathbb{E}[\langle \nabla f(x), u \rangle u] = \nabla f(x)\,. \tag{4.4}$$

**Applicability to Local Learning** In the context of local learning, this opens the way to obtaining improved estimates of the End-to-End gradient solely via forward passes. Indeed, we can now compute with a forward pass based on a guess $u_j$ the directional derivative given by

$$\langle F_j, u_j \rangle u_j. \tag{4.5}$$

While several "gradient guesses" could be used, a relatively natural candidate to the gradient "guess" is thus

$$u_j = \frac{\nabla(\ell_j \circ g_j)}{\|\nabla(\ell_j \circ g_j)\|}\,.$$

In this context, Equation 4.5 serves as the best linear approximation of $F_j$ along the given direction $u_j$. Despite extensive experimentation, we were unable to identify a setting where this approximation proved advantageous. As an alternative approach, we conducted an ablation study to evaluate the impact of various components.

With two students, Louis Fournier and Dr. Stéphane Rivaud, we carried out a comprehensive analysis of this method across multiple datasets and algorithmic variations which we discuss next.

**Ablation experiments to understand the limit of Forward Gradient** A sample of result can be seen in Table 4.2, obtained from the numerous experiments performed in Fournier et al. [2023]. We studied several potential guesses $u_j$ on the CIFAR-10 and ImageNet-32 datasets, using a layer-wise split. Three classes were used: the Local refers to a supervised auxiliary Neural Network, potentially a Convolutional Neural Network, or a Multi-Layer Perceptron or a Linear Layer. Second, we also investigated if such supervision could lead to

anything effective if we make this random neural network random (in practice, we simply reinitialize at every iterations the model) - this approach is very similar to consider a finite width random Neural Tangent Kernel (see Jacot et al. [2018]. Thirdly, we proposed to use random guesses as in Silver et al. [2021], Satorras et al. [2021]. The weight or activation depends if the correction has been applied on the gradients computed w.r.t. the weights or the activations. Without any surprise, the more prior we incorporate, the better are the results. The Multi-Layer Perceptron appears to be quite competitive, yet the best performance on ImageNet 32are obtained with a gradient weight estimation via a Convolutional Neural Network.

**The limit of Local Losses** While local losses allow a great amount of parallellization, there are at least two outcomes: **(a)** local losses still lack the gradient estimate quality of end-to-end **(b)** sometimes, it is difficult to infer an appropriate local-losses: in the case of generative model, finding a good layerwise criterium is an open question.

# Chapter 5

# Data-parallelism: Asynchronous Decentralized Algorithms

This chapter is based on the works:

- Adel Nabli and Edouard Oyallon. Dadao: Decoupled accelerated decentralized asynchronous optimization. In *International Conference on Machine Learning*, pages 25604–25626. PMLR, 2023b which is discussed in Section 5.3,
- Adel Nabli and Edouard Oyallon. Decentralized asynchronous optimization with dadao allows decoupling and acceleration. *preprint*, 2023a is an extension which studies particularly graphs and discussed in Section 5.2,
- Adel Nabli, Eugene Belilovsky, and Edouard Oyallon. $\mathbf{A}^2\mathbf{CiD}^2$: Accelerating asynchronous communication in decentralized deep learning. *Advances in Neural Information Processing Systems*, 2023 is an algorithm which focuses on acceleration communications for Deep Learning settings, discussed in Section 5.4.

The main student involved is Adel Nabli.

The over goal of this chapter is to propose asynchronous procedures to solve problems with a separable structure

$$\inf_{x\in\mathbb{R}^d} \sum_{i=1}^{n} f_i(x)\,, \tag{5.1}$$

where each $f_i : \mathbb{R}^d \to \mathbb{R}$ is a function with some regularity, e.g., $L$-smooth and $\mu$-strongly convex. Solving Equation 5.1 via a first-order method can be computationally intense, as it requires calling $n$ data queries or gradient oracles per

gradient step: this is why stochastic gradient procedures are employed [Bottou and Bousquet, 2007]. Typically, a gradient estimate is given by $\nabla f_{\mathbf{i}}(x)$ where $\mathbf{i}$ follows a uniform distribution on $\{1, .., n\}$. In this case, it is an unbiased estimate of $\frac{1}{n}\sum_{i=1} \nabla f_i(x)$.

**Optimization in decentralized context** In a typical decentralized context, one has a network of $n$ nodes, whose topology is determined by a (potentially directed) graph with edges given by $\mathcal{E} \subset \{1, ..., n\} \times \{1, ..., n\}$. In this case, one often introduces the concept of connectivity matrice $[\lambda_{ij}]_{ij}$, where $\lambda$ is a $n \times n$ matrix so that $\lambda_{ij} = 0$ iff $(ij) \notin \mathcal{E}$.

Each node has a local memory, and has a gradient oracle access to $\nabla f_i$. In most of the decentralized works, the communication corresponds to a global communication step, meaning that all edges involved simultaneously spike during a communication.

**Local SGD and mini-batch SGD** While mini-batch SGD relies on a an aggregation of results on a central worker, local SGD [Stich, 2018, Woodworth et al., 2020] takes the approach to perform more local steps in order to avoid communication overhead. Yet, local SGD does not enjoy a variance reduction as mini-batch SGD.

**Counting communication.** While most decentralized works count the number of communication rounds, this metric does not make sens in an asynchronous setting. Instead, we propose to count the expected number of edges which are spiking (i.e., are activated), as in Boyd et al. [2006b].

**Toward asynchrony: stochastic algorithms in decentralized context?** In Table 5.1, I summarize the lower bounds, that I am aware of, for decentralized and centralized algorithms with a finite sum structure (obtained from Scaman et al. [2017] and Woodworth and Srebro [2016]). In the case of the centralized setting, local SGD could allow to reduce the communication rate (i.e., less than one communication per gradient), yet I am unaware of lower bounds. Similarly, the limit of stochastic algorithms in the decentralized learning is slightly unclear to me. In the centralized setting, stochastic algorithm lower the number of gradient oracle calls by using variance reduction: however, it is slightly unclear how to apply this technique in a decentralized algorithm optimal in communication, where there is no common buffers to share information... Asynchrony brings an additional challenge, as the gradients are potentially neither computed w.r.t. the same time, nor the same parameters. Filling this table could correspond to future exciting research directions.

**Discussion and organization of this chapter** The first Section 5.1 of this chapter will discuss randomized gossip algorithm, which highly motivates the concept of asynchronous algorithm and this research. We explain in Section 5.2 how to further accelerate those algorithms. Based on the very exciting

Table 5.1: Summary of lower-bounds for communication and gradient oracle costs for $L$-smooth and $\mu$-strongly convex functions, to solve Equation (5.1). Here, $n$ is the number of workers (and the length of the finite sum), while $\gamma$ is the spectral gap. Note that for a star-graph, $\gamma = n$.

| | Decentralized | | Centralized | |
|---|---|---|---|---|
| | Stochastic | Deterministic | Stochastic | Deterministic |
| Grad. | **?** | $\log \frac{1}{\epsilon} n \sqrt{\frac{L}{\mu}}$ | $\log \frac{1}{\epsilon}(n + \sqrt{n\frac{L}{\mu}})$ | $\log \frac{1}{\epsilon} n \sqrt{\frac{L}{\mu}}$ |
| Comm. | **?** | $\lvert\mathcal{E}\rvert \sqrt{\gamma} \log \frac{1}{\epsilon} \sqrt{\frac{L}{\mu}}$ | **?** | **?** |

works Kovalev et al. [2021a], Even et al. [2021], we designed DADAO in Nabli and Oyallon [2023b], as it is an example of asynchronous algorithm which gives a practical and theoretical benefits, and Section 5.4 explains to apply this technique to a Deep Learning setting.

**Reminders on Stochastic Differential Equation** For $\Xi$ a measurable space, let $x : \mathbb{R}_+ \times \Xi \to \mathbb{R}^d$ be a random process adapted to the filtration $\{\mathcal{F}_t\}_t$ generated by a Poisson Process $N_t$, whose dynamic follows (with a small abuse in notations)

$$dx_t = f(x_t)dt + g(x_t)dN_t\,. \tag{5.2}$$

In this case, for any smooth potential $V : \mathbb{R}^d \to \mathbb{R}^+$, we have via Itô's formula [Karatzas and Shreve, 1991]

$$V(x_t) - V(x_0) = \int_0^{\mathsf{T}} \langle \nabla V(x_u), f(x_u)\rangle + \big(V(x_u + g(x_u)) - V(x_u)\big)\, du + M_t\,, \tag{5.3}$$

where $M_t \triangleq \int_0^{\mathsf{T}} \big(V(x_u + g(x_u)) - V(x_u)\big)dN_u - \int_0^{\mathsf{T}} \big(V(x_u + g(x_u)) - V(x_u)\big)du$ is a martingale adapted to the filtration of $x_t$, meaning that for $s \leq t$, we have

$$\mathbb{E}[M_t | \mathcal{F}_s] = M_s\,. \tag{5.4}$$

In particular, this implies that $\mathbb{E}[M_t] = 0, \forall t$. The proofs of this chapter are almost exclusively based on Itô's formula. In other words: we have a discrete stochastic algorithm, combined with a continuous proof which is an extremely elegant finding of Even et al. [2021].

## 5.1 Basic of asynchrony: graphs and gossips

### 5.1.1 Non-accelerated setting

Let us consider a directed graphs with weights $\lambda_{ij} > 0$ iff $(ij) \in \mathcal{E}$. We introduce its symmetric Laplacian given by

$$\Lambda = \sum_{(ij)\in\mathcal{E}} \lambda_{ij} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^{\mathsf{T}}\,. \tag{5.5}$$

It is pretty easy to see that $\mathbf{1}$ is in the kernel of $\Lambda$ and that if the graph is connected, then the kernel is reduced to span($\mathbf{1}$). The major quantity used to analyze gossip algorithm rates of convergence is the inverse of the smallest eigenvalue of matrix in Equation 5.5, given by

$$\chi_1 \triangleq \sup_{\|x\|=1, x \perp \mathbf{1}} \frac{1}{x^\mathsf{T} \Lambda x} \leq \infty \,. \tag{5.6}$$

If the graph is connected, then $\chi_1 < \infty$. A consensus optimization problem consists in finding $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ only with pair-wise, independent communications. A standard algorithm obtained by Boyd et al. [2006b] consists simply in studying the dynamic

$$dx_i(t) = -\frac{1}{2} \sum_{(ij) \in \mathcal{E}} (x_i(t) - x_j(t)) dN_{ij}(t) \,, \tag{5.7}$$

where $N_{ij}$ is a Poisson Process with intensity $\lambda_{ij}$. In this case, noting that the dynamic also writes $d\mathbb{E}[x_t] = -\frac{1}{2}\Lambda \mathbb{E}[x_t] dt$, and, we get

$$\mathbb{E}[\|\pi x_t\|^2] \leq e^{-\frac{t}{2\chi_1}} \|\pi x_0\|^2 \,.$$

In this context, to reach a specific accuracy $\epsilon$, the total number of activated edges is proportional to $\chi_1 \mathrm{Tr}\ \Lambda \log \frac{1}{\epsilon}$. Interestingly, this value is invariant by rescaling of $\Lambda$ by a constant.

**Comparison to the synchronous setting.** For the sake of comparison, consider a synchronous gossip algorithm that updates its state $x_{t+1}$ as follows

$$x_{t+1} = D x_t - \alpha \Lambda x_t \,.$$

Here, $D$ is a diagonal matrix, ensuring that each update is local, and $\alpha$ is a non-negative constant. Basic analysis shows that $D$ should be the identity matrix $\mathbf{I}$, leading to a constraint on $\alpha$ given by

$$0 \preccurlyeq \alpha \Lambda \preccurlyeq 2\mathbf{I},$$

and thus $\alpha \leq \frac{2}{\|\Lambda\|}$. This condition results in the total number of activated edges being proportional to $|\mathcal{E}| \|\Lambda\| \chi_1 \log \frac{1}{\epsilon}$, which also remains invariant under scaling. Furthermore, $\mathrm{Tr}\ \Lambda \leq n\|\Lambda\| \leq |\mathcal{E}| \|\Lambda\|$, proving that asynchronous gossip algorithms necessarily require less spiking edges compared to their synchronous counter-part. This inequality can be large, as shown for the star graph.

Finally, let $\gamma \triangleq \|\Lambda\| \chi_1$, which is also referred to as the spectral gap [Scaman et al., 2017] and guide the convergence rate of decentralized algorithms. Note that if the sum of the weights of each node is 1, then $\mathbf{I} - \Lambda$ is a contractive matrix due to the Perron-Frobenius theorem with spectral radius 1. In other words, $\|\Lambda\|$ is simply to compute, and this is a standard assumption [Boyd et al., 2006b].

The fact that stochastic communications lead to faster algorithms is not surprising: stochastic algorithms demonstrate superiority over deterministic ones [Woodworth and Srebro, 2017, Defazio et al., 2014].

**Time-varying setting.** It is possible to use some time-varying weights $\lambda_{ij}(t)$, The current analysis then considers that the communication speed is guided by the most poorly connected graph, and they thus rely on

$$\chi_1^* \triangleq \sup_{t>0} \sup_{\|x\|=1, x\perp \mathbf{1}} \frac{1}{x^\mathsf{T} \Lambda(t) x} \,. \tag{5.8}$$

### 5.1.2 The graph resistance

For a graph with Laplacian $\Lambda$, we remind that its Moore-Penrose pseudo inverse $\Lambda^+$ is also a symmetric positive semi-definite matrix. We define the resistance $R_{ij}$ between two nodes $i, j$ (not necessarily connected by an edge) via

$$R_{ij} = (\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} \Lambda^+ (\mathbf{e}_i - \mathbf{e}_j) \,. \tag{5.9}$$

The maximal resistance of $\Lambda$ is then given by

$$\chi_2 \triangleq \sup_{(ij)\in\mathcal{E}} R_{ij} \,. \tag{5.10}$$

We now discuss the interpretation of these quantities. The reason we will get acceleration is that we have clearly

$$\chi_2 \leq \chi_1 \,. \tag{5.11}$$

This quantity can be substantially smaller as discussed later in Table 5.2.

**First parallel: electricity.** The graph resistance allows us to draw an exact parallel with electrical resistance. Setting $r_{ij} = \frac{1}{\lambda_{ij}}$, Ohm laws and Kirchhoff laws apply [Chandra et al., 1996] and the resistance between two nodes of the network is exactly the electrical resistance. It also means that symmetry, as used in electricity, can be used to compute the resistance. The resistance of a graph, thanks to the rich invariant geometric it encodes, has wide applications in crystallography [Klein and Randić, 1993].

**Second parallel: commute time.** Another interpretation comes from random walks with reversible Markov chains: let us show that the comute time between two nodes $i, j$ of a random walk is actually the graph resistance between $i, j$, up to a multiplicative constant. To stress the simplicity of our framework, we give a simple proof. Let us assume that $\lambda_{ij} = \lambda_{ji}$. Indeed, let us write $p(i|j) = \frac{\lambda_{ij}}{\sum_i \lambda_{ij}}$ the reversible transition kernel from $i$ to $j$. Let $H_{ij}$ be the hitting time of $j$ starting at $i$, which is thus in expectation defined as $H_{ij} = \mathbb{E}[\arg\inf_t \{x_t = j\} | x_0 = i]$, and $H_{ii} = 0$ by convention. Stated simply, we have the basic recursion, by considering the path $i \to j$ and $i \to k \to j$ for $k \in \mathcal{N}(i)$ so that

$$H_{ij} = 1 + \sum_{k\in\mathcal{N}(i)} p(k|i) H_{kj} \,. \tag{5.12}$$

One would get with a bit of work a similar equation satisfied by $R_{ij}$, however there is no unicity to the solutions. Letting $d = [\sum_k \lambda_{ik}]_i \in \mathbb{R}^n$ and $D$ be the corresponding diagonal matrix, then Equation (5.12) also rewrites

$$\forall j \neq i, 1 = [\frac{1}{2} D^{-1} \Lambda H]_{ij} \text{ or } 2e_i^\mathsf{T} d = e_i^\mathsf{T} \Lambda H e_j \,. \tag{5.13}$$

In other words, as $\Lambda \mathbf{1} = 0$, this writes

$$\frac{1}{2}\Lambda H e_j = d - (d^\mathsf{T}\mathbf{1})e_j \text{ or } \frac{1}{2} H e_j = \Lambda^+(d - (d^\mathsf{T}\mathbf{1})e_j) + ((He_j)^\mathsf{T}\mathbf{1})\mathbf{1} \,. \tag{5.14}$$

And since $e_j^\mathsf{T} H e_j = 0$, this also writes

$$\frac{1}{2} H e_j = \Lambda^+(d - (d^\mathsf{T}\mathbf{1})e_j) - (e_j^\mathsf{T}\Lambda^+(d - (d^\mathsf{T}\mathbf{1})e_j))\mathbf{1} \,. \tag{5.15}$$

And at the end, we get

$$e_i^\mathsf{T} H e_j = 2(e_i - e_j)^\mathsf{T}\Lambda^+(d - (d^\mathsf{T}\mathbf{1})e_j) \,. \tag{5.16}$$

Combining, we have

$$H_{ij} + H_{ji} = 2\mathrm{Tr}\,\Lambda(e_i - e_j)\Lambda^+(e_i - e_j) = 2\mathrm{Tr}\,\Lambda R_{ij} \,. \tag{5.17}$$

### 5.1.3 Acceleration

**Tchebychev acceleration** As explained in Section 5.1, cascading several steps of gossip communication allow to obtain a contracting communication operator: $\|(\mathbf{I} - \alpha\Lambda)^{\lfloor \mathcal{O}(\gamma)\rfloor} x\| \leq \frac{1}{2}\|\pi x\|$. In this case, about $\gamma$ gossip steps are needed, and this contracting step is key to most of decentralized works [Scaman et al., 2017, Kovalev et al., 2022]. However, it is possible to compute a much more efficient contraction, using Chebychev polynomial of order $n$, written $P_n$. In this case, replacing $(\mathbf{I} - \alpha\Lambda)^{\lfloor \mathcal{O}(\gamma)\rfloor}$ by $P_n(\mathbf{I} - \alpha\Lambda)$, where $n = \lfloor\sqrt{\gamma}\rfloor$, one also gets $\|P_n(\mathbf{I} - \alpha\Lambda)x\| \leq \frac{1}{2}\|\pi x\|$. A Chebychev polynomial $P_n$ can be computed in roughly $n$ communications, which reduces the number of global gossip steps from $\gamma$ to $\sqrt{\gamma}$ which is a significant improvement. However, it has at least two issues: first, it requires a synchronous rounds of communications where each nodes spike. Second, it is also unclear how to accelerate randomly sampled and/or time-varying graphs.

**Accelerating Randomized gossip** In the cas of accelerated gossip, Even et al. [2021] showed that the following dynamic

$$\begin{aligned} dx_t &= \alpha(\tilde{x}_t - x_t)dt - \beta \sum_{(i,j)\in\mathcal{E}} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} x_t dM_{ij}(t) \\ d\tilde{x}_t &= \alpha(x_t - \tilde{x}_t)dt - \tilde{\beta} \sum_{(i,j)\in\mathcal{E}} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\mathsf{T} x_t dM_{ij}(t) \,. \end{aligned} \tag{5.18}$$

Under appropriate parameters, which depends on $\chi_1, \chi_2$, this dynamic leads to an accelerated rate of $\operatorname{Tr}\Lambda\sqrt{\chi_1\chi_2}\log\frac{1}{\epsilon}$ communications in expectations. It is possible to show that $\lambda_{ij}R_{ij} \leq 1$ [Nabli and Oyallon, 2023b]. Thus if $\lambda_{ij} = 1$, we note that this leads to a number of exepected spiking edges of $|\mathcal{E}|\sqrt{\chi_1}\log\frac{1}{\epsilon}$, which improves over the synchronous accelerated algorithm which will lead to $|\mathcal{E}|\sqrt{\chi_1\|\Lambda\|}\log\frac{1}{\epsilon}$ activated edges.

**Open questions** In the time varying setting, Metelev et al. [2023] constructs a sequence of time varying graphs, for which no acceleration is possible. This strongly suggests that the same property should hold for randomized accelerated gossips.

## 5.2 Faster gossips

We now discuss a simple relaxation that we derived in Nabli and Oyallon [2023a]. For the sake of accelerating further gossip algorithms, we considered the following problem

$$\begin{aligned} \text{minimize} \quad & \operatorname{Tr}\,\Lambda(\lambda)\sqrt{2\chi_1[\Lambda(\lambda)]\chi_2[\Lambda(\lambda)]} \\ \text{subject to} \quad & \lambda \in \mathcal{C} \cap \mathbb{R}_+^{\mathcal{E}}\,. \end{aligned} \tag{5.19}$$

where we stress here the dependency of $\chi_1(\Lambda(\lambda))$ and $\chi_2(\Lambda(\lambda))$ in $\lambda$, and $\mathbb{R}_+^{\mathcal{E}} = \{\lambda_{ij} \geq 0, \lambda_{ij} = 0 \text{ if } (ij) \notin \mathcal{E}\}$ and $\mathcal{C}$ is a constraint on the conneectivity. For the sake of simplicity, we will consider the contraint $\operatorname{Tr}\Lambda = 1$ to have a global bandwidth condition. In this case, we can show that

**Lemma 1** (log-convexity lemma, from Nabli and Oyallon [2023a])**.** *Fix $\|u\| = 1$, $u \perp \mathbf{1}$ and let $\lambda = (\lambda_{ij}) \in \mathbb{R}_+^{n^2}$ such that $\Lambda(\lambda) = \sum_{1\leq i,j\leq n}\lambda_{ij}(e_i - e_j)(e_i - e_j)^\mathsf{T}$, and let*

$$\psi(\lambda, u) = u^\mathsf{T}\Lambda(\lambda)^+ u\,.$$

*Then, $\lambda \to \psi(\lambda, u)$ is log-convex on $\mathbb{R}_+^{n^2}$. Furthermore, $\lambda \to \psi(\lambda, u)$ is strictly convex on the convex set $\{\lambda, \chi_1[\Lambda(\lambda)] < \infty\}$ of the connected graphs.*

While important, this Lemma does not imply that $\lambda \to \chi_2[\Lambda(\lambda)]$ is convex. Indeed, the supremum is taken over a set of edge *dynamically* defined through $\lambda$, $\mathcal{E}(\lambda) \triangleq \{(i,j), \lambda_{ij} > 0\}$ which makes it non-convex: an optimal solution might lead to removing some edges from the graph and consider the maximum effective resistance of this sub-graph. In fact, it also shows that contrary to the situation of minimizing $\chi_1[\Lambda(\lambda)]$ in Boyd et al. [2006a], there might be no unique minimum. This forces us to slightly relax this condition by taking the supremum over a predefined *fixed* set of edges, and we introduce for any $\lambda \in \mathbb{R}_+^{\mathcal{E}}$

$$\chi_2^{\mathcal{E}}(\lambda) \triangleq \frac{1}{2}\max_{(i,j)\in\mathcal{E}}(e_i - e_j)^\top\Lambda^+(\lambda)(e_i - e_j)\,.$$

This function is convex, and we note that by definition $\chi_2^{\mathcal{E}}(\lambda) \geq \chi_2[\Lambda(\lambda)]$. We have the following Lemma, obtained using Lemma 1, that shows that relaxing $\chi_2$ with $\chi_2^{\mathcal{E}}$ leads to a convex problem:

**Lemma 2** (Strict convexity of the relaxed communication rate, from Nabli and Oyallon [2023a])**.** *Fix a directed graph $\mathcal{E}$ so that $\bar{\mathcal{E}}$ is connected. Then, $\lambda \to \sqrt{\chi_1[\Lambda(\lambda)]\chi_2^{\mathcal{E}}(\lambda)}$ is strictly convex on $\mathbb{R}_+^{\mathcal{E}}$.*

Table 5.2: Communication rates using uniform weights and optimal rates under the condition $\mathcal{C}_1$. To reach $\epsilon$-precision, multiply the complexities by a $\mathcal{O}(\log(\frac{1}{\epsilon}))$.

| Type | Renormalization by the degree | | | | | Optimal weights | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Graph | Star | Line | Complete | $d$-grid | Barbell | Star | Line | Complete | $d$-grid | Barbell |
| Sync Gossip | $n^2$ | $n^3$ | $n^2$ | $n^{1+\frac{2}{d}}$ | $n^4$ | $n^2$ | $n^3$ | $n^2$ | $n^{1+\frac{2}{d}}$ | $n^2$ |
| Acc. Sync. Gossip | $n^{\frac{3}{2}}$ | $n^2$ | $n^2$ | $n^{1+\frac{1}{d}}$ | $n^3$ | $n^{\frac{3}{2}}$ | $n^2$ | $n^2$ | $n^{1+\frac{1}{d}}$ | $n^2$ |
| Async. Gossip | $n$ | $n^3$ | $n$ | $n^{1+\frac{2}{d}}$ | $n^3$ | $n$ | $n^3$ | $n$ | $n^{1+\frac{2}{d}}$ | $n$ |
| Acc. Async. Gossip | $n$ | $n^2$ | $n$ | $n^{1+\frac{1}{d}}$ | $n^{5/2}$ | $n$ | $n^2$ | $n$ | $n^{1+\frac{1}{d}}$ | $n$ |

In this case, it can be shown that minimizing Equation (5.2) is equivalent to minimizing the following SDP

$$\begin{aligned}
\text{minimize} \quad & t_1 + t_2 \\
\text{subject to} \quad & \lambda \in u\mathcal{C}, u \geq 1, \lambda_{ij} \geq 0 \; \forall (i,j) \in \mathcal{E} \\
& \Lambda = \textstyle\sum_{ij} \lambda_{ij} (e_i - e_j)(e_i - e_j)^\mathsf{T} \\
& \begin{pmatrix} \Lambda & u(e_i - e_j) \\ u(e_i - e_j)^\mathsf{T} & t_1 \end{pmatrix} \succcurlyeq 0 \quad \forall (i,j) \in \mathcal{E} \\
& \begin{pmatrix} \Lambda & \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\mathsf{T} \\ \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\mathsf{T} & t_2 \mathbf{I} \end{pmatrix} \succcurlyeq 0 \,.
\end{aligned}$$

Some results are provided in Figure 5.2, which indicate substantial improvement thanks to our method compared to predefined weights, and Table 5.2 summarizes some thereotical rates and a comparison with synchronous algorithms.

## 5.3 DADAO: Decoupled Accelerated Decentralized Asynchronous Optimization

The goal of this joint work with Adel Nabli, was to propose a theoretical framework for asynchrony, published at ICML [Nabli and Oyallon, 2023b]. The basis of DADAO follows an observation stated in Kovalev et al. [2020, 2021a], Salim et al. [2022], Hendrikx [2022], that the Objective (5.1) can be rewritten equivalently, for $0 < \nu < \mu$,

$$\inf_{x \in \mathbb{R}^{n \times d}} \sup_{\substack{y \in \mathbb{R}^{n \times d} \\ z \in \mathbb{R}^{n \times d}}} \sum_{i=1}^{n} f_i(x_i) - \frac{\nu}{2}\|x\|^2 - \langle x, y \rangle - \frac{1}{2\nu}\|\pi z + y\|^2. \tag{5.20}$$

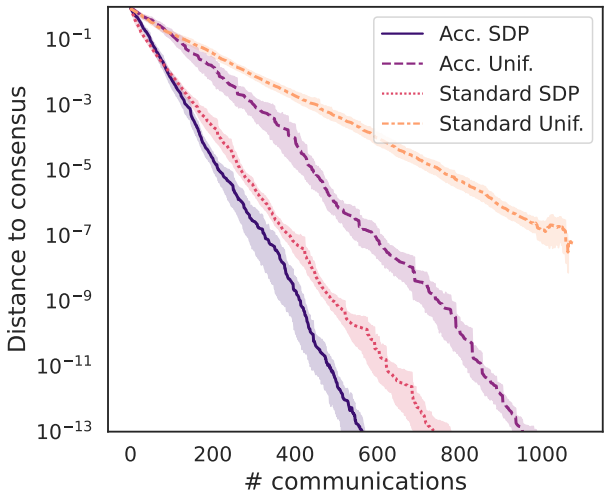


Figure 5.1: Comparison between Standard Randomized Gossip, Accelerated Randomized Gossip, using both a Gossip matrix optimized via SDP and the one with uniform edge weight. Average of 50 runs on random geometric graphs of size, from left to right, for $n = 12$ nodes. Note that, systematically, our method leads to a substantial speed improvement.

Studying the saddle points of this problem have led to propose the following dynamic, where $\mathbf{N}(t)$ is a Poisson Process, $M_{ij}(t)$ are Poisson Processes determined by $\Lambda$, and $\eta, \tilde{\eta}, \gamma, \tilde{\gamma}, \alpha, \tilde{\alpha}, \beta, \tilde{\beta}, \delta, \tilde{\delta}$ are scalar parameters

$$
\begin{aligned}
dx_t &= \eta(\tilde{x}_t - x_t)dt - \gamma(\nabla f(x_t) - \nu x_t - \tilde{y}_t)\, d\mathbf{N}(t) \\
d\tilde{x}_t &= \tilde{\eta}(x_t - \tilde{x}_t)dt - \tilde{\gamma}(\nabla f(x_t) - \nu x_t - \tilde{y}_t)\, d\mathbf{N}(t) \\
d\tilde{y}_t &= -\theta(y_t + z_t + \nu\tilde{x}_t)dt + (\delta + \tilde{\delta})(\nabla f(x_t) - \nu x_t - \tilde{y}_t)d\mathbf{N}(t) \\
dy_t &= \alpha(\tilde{y}_t - y_t)dt \\
dz_t &= \alpha(\tilde{z}_t - z_t)dt - \beta \sum_{(i,j)\in\mathcal{E}} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^{\mathsf{T}}(y_t + z_t)dM_{ij}(t) \\
d\tilde{z}_t &= \tilde{\alpha}(z_t - \tilde{z}_t)dt - \tilde{\beta} \sum_{(i,j)\in\mathcal{E}} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^{\mathsf{T}}(y_t + z_t)dM_{ij}(t)\,.
\end{aligned}
\tag{5.21}
$$

This dynamic shares some similarity with Kovalev et al. [2021a], as it relies also on the saddle points of Equation (5.20). However, contrary to Kovalev et al. [2021a], we do not employ a Forward-Backward algorithm, which requires both an extra-inversion step and additional regularity on the considered proximal operator. Not only does this condition not hold in this particular case, but this is not desirable in a continuized framework where iterates are not ordered in a predefined sequence and require a local descent at each instant. Another

major difference is that no Error-feedback is required by our approach, which allows unlocking asynchrony while simplifying the proofs and decreasing the required number of communications. It is worth noting the non-trivial term $(y_t + z_t + \nu\tilde{x}_t)dt$ of line 3 of Equation (5.21) which appears crucial to synchronize the couples of variables $\{x_t, y_t, \tilde{x}_t, \tilde{y}_t\}$ and $\{z_t, \tilde{z}_t\}$. This mixing allows the decoupling and the communication acceleration: indeed, without it, there would be no interaction between communications (lines 4-6 of Equation (5.21)) and computations (lines 1-3 of Equation (5.21)). Note also that no global "ping" routines, as in gradient tracking is required by this dynamic: it is a fully asynchronous algorithm. The following theorem allows us to control the rate of convergence of the dynamic of Equation (5.21):

**Theorem 5.3.1** (Adapted from Nabli and Oyallon [2023b])**.** *Assume each $f_i$ is $\mu$-strongly convex and $L$-smooth. If $\Lambda$ is connected, then there exists some parameters for the dynamic Eq.* (5.21)*, such that for any initialization $x_0 \in ker(\pi)$, and $\tilde{x}_0 = x_0, y_0 = \tilde{y}_0 = \nabla f(x_0) - \frac{\mu}{2}x_0, z_0 = \tilde{z}_0 = -\pi y_0$, we get for $t \in \mathbb{R}^+$*

$$\mathbb{E}[\|x_t - x^*\|^2] \leq (\frac{1}{2} + \frac{23}{8}\frac{L}{\mu} + 2\frac{L^2}{\mu^2})\|x_0 - x^*\|^2 e^{-\frac{t}{16\sqrt{2}\sqrt{\chi_1\chi_2}}\sqrt{\frac{\mu}{L}}} .$$

**Complexity analysis:** In this case, if the algorithm runs for a time $t$, the expected number of gradient oracle calls is thus $nt$ and the expected number of communication is $\mathrm{Tr}\,\Lambda t$. In other words, to reach precision $\epsilon$, one needs $n\sqrt{\chi_1\chi_2\frac{L}{\mu}}\log\frac{1}{\epsilon}$ gradient oracle calls and $\mathrm{Tr}\,\Lambda\sqrt{\chi_1\chi_2\frac{L}{\mu}}\log\frac{1}{\epsilon}$ communications. In other words, replacing $\Lambda$ by $\tilde{\Lambda} = \sqrt{\chi_1\chi_2}\Lambda$ (which leads to accelerating the number of communication per unit time), leads to $n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$ gradient oracle calls and $\mathrm{Tr}\,\Lambda\sqrt{\chi_1\chi_2\frac{L}{\mu}}\log\frac{1}{\epsilon}$ communications.

**Open question and problems:** The number of gradient calls for DADAO, being a stochastic algorithm, to reach $\epsilon$-precision corresponds to $n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$. At the same time, Woodworth and Srebro [2017] have pointed out that the rate of $(n + \sqrt{n\frac{L}{\mu}})\log\frac{1}{\epsilon}$ oracle calls to reach an $\epsilon$ level of precision, is a lower-bound for stochastic algorithms. However, there might be two issues to overcome to obtain this rate.

The first aspect pertains to the absence of shared variables for result aggregation. For variance reduction techniques, a common practice involves employing a centralized buffer for aggregating variables or gradients. The absence of such a shared variable questions the possibility of DADAO to reach this rate.

The second issue relates to worker synchronization. In DADAO, synchronization among workers is generally avoided, allowing individual workers to make independently compute gradients and communicate. This feature is at odds with traditional variance reduction strategies in convex optimization, where

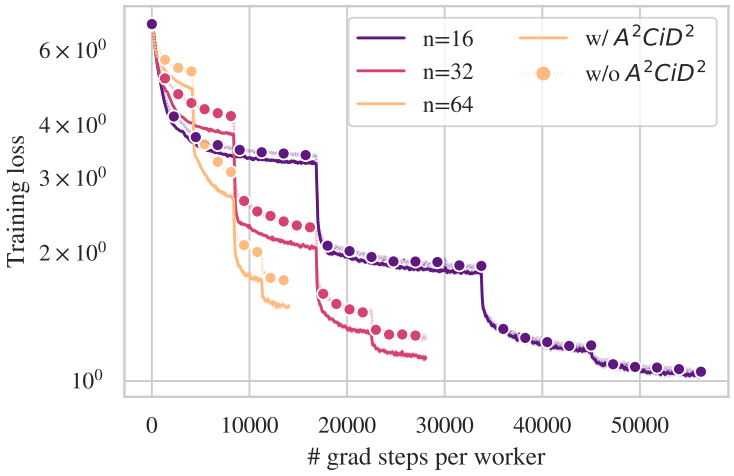


Figure 5.2: Training loss for ImageNet using 128 batch size, with an equal number of communications and computations per worker. We display the training loss for various number of workers (up to 64), using $\mathbf{A^2CiD^2}$, for the challenging ring graph.

Table 5.3: Training times on CIFAR10 ($\pm$ 6s).

| | $n$ | 4 | 8 | 16 | 32 | 64 |
|---|---|---|---|---|---|---|
| Ours | $t$ (min) | 21.4 | 10.8 | 5.7 | 3.2 | 1.9 |
| AR | $t$ (min) | 21.9 | 11.1 | 6.6 | 3.2 | 1.8 |

simultaneous computation and aggregation are usually necessary for effective reduction of variance. Thus, reconciling the optimality of DADAO's complexity, particularly in a stochastic and decentralized context, is an open question.

## 5.4 $\mathbf{A^2CiD^2}$: Accelerating Asynchronous Communication in Decentralized Deep Learning

A major drawback of DADAO is its non applicability to Deep Learning settings, as there is no equivalent of duality in Deep Learning: the formulation Equation (5.20) does not hold. However, in most of settings, communication is a linear operation, which in fact results from a convex problem. A major limit of Chebyshev acceleration is that it requires synchronous, sequential rounds, and that nodes simultaneous spike. Consequently, this approach is not applicable to an asynouc Deep Learning setting. The objective of Nabli et al. [2023] is to accelerate approaches like Zhang et al. [2015], Blot et al. [2016], Daily et al. [2018], Lian et al. [2018], Assran et al. [2019], which proposes a training algo-

rithm whose particularity is to simultaneously perform asynchronous gradients and communication in parallel.

**Dynamic.** In $\mathbf{A^2CiD^2}$, we consider the following dynamic, using similar notations to Section 5.3, for a setting with $n$ workers

$$\begin{aligned} dx_t^i &= \eta(\tilde{x}_t^i - x_t^i)dt \\ &\quad - \alpha \sum_{j,(i,j)\in\mathcal{E}} (e_i - e_j)(e_i - e_j)^\mathsf{T} x_t dM_{ij}(t) \\ &\quad - \gamma \int_\Xi \nabla F_i(x_t^i, \xi_i)\, dN_i(t, \xi_i)\,, \qquad (5.22) \\ d\tilde{x}_t^i &= \eta(x_t^i - \tilde{x}_t^i)dt \\ &\quad - \tilde{\alpha} \sum_{j,(i,j)\in\mathcal{E}} (e_i - e_j)(e_i - e_j)^\mathsf{T} x_t dM_{ij}(t) \\ &\quad - \gamma \int_\Xi \nabla F_i(x_t^i, \xi_i)\, dN_i(t, \xi_i)\,. \qquad (5.23) \end{aligned}$$

Here, $\Xi$ is a measurable space and $dN_i(t, \xi_i)$ is a Poisson measure with intensity $dt \otimes d\mathcal{P}(\xi)$ defined over $\mathbb{R}_+ \times \Xi$, while $M_{ij}(t)$ is as above. If $\eta = 0$, this dynamic is equivalent to a standard Stochastic Push algorithm, where spikes are modeled by Poisson processes. However, by adding a second momentum variable, we are able accelerate the communication. We will now consider two generic assumptions obtained from Koloskova et al. [2020], which allow to specify our lemma to convex and non-convex settings. Note that the non-convex Assumption 5.4.2 generalizes the assumptions of Lian et al. [2018], by taking $M = P = 0$.

**Assumption 5.4.1** (Strongly convex setting)**.** *Each $f_i$ is $\mu$-strongly convex and $L$-smooth, and:*

$$\frac{1}{n}\sum_{i=1}^n \mathbb{E}_{\xi_i}[\|\nabla F_i(x, \xi_i) - \nabla f_i(x)\|^2] \leq \sigma^2 \text{ and } \frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x^*) - \nabla f(x^*)\|^2 \leq \zeta^2\,.$$

and

**Assumption 5.4.2** (Non-convex setting)**.** *Each $f_i$ is $L$-smooth, and there exists $P, M > 0$ such that:*

$$\forall x \in \mathbb{R}^d, \frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x) - \nabla f(x)\|^2 \leq \zeta^2 + P\|\nabla f(x)\|^2\,,$$

*and,*

$$\forall x_1, ..., x_n \in \mathbb{R}^d, \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{\xi_i}\|\nabla F_i(x_i, \xi_i) - \nabla f_i(x_i)\|^2 \leq \sigma^2 + \frac{M}{n}\sum_{i=1}^n \|\nabla f_i(x_i)\|^2\,.$$

Now, we were able to obtain the following rate, which are further compared in Table 5.4:

**Theorem 5.4.3** (Convergence guarantees, adapted from Nabli et al. [2023])**.** *Assume that $\{x_t, \tilde{x}_t\}$ follow the dynamic Equation (5.22) and the network connectivity is given by $\chi_1, \chi_2$ as in Section 5.3. Assume that $\mathbf{1}\bar{x}_0 = x_0 = \tilde{x}_0$ and let $T$ the total running time. Then, under Assumptions 5.4.1 and 5.4.2, depending if $\chi = \chi_1$ or $\chi = \sqrt{\chi_1\chi_2}$, there exists some parameters $\eta, \alpha$ and a step size such that*

- *(**strong-convexity**) if each function is $\mu$-strongly convex and $L$-smooth, then $\gamma < \frac{1}{16L(1+\chi)}$ and*

$$\mathbb{E}\|\bar{x}_T - x^*\|^2 = \|\bar{x}_0 - x^*\|^2 e^{-T\frac{\mu}{128\chi L}} + \frac{64}{\mu^2 T}(\sqrt{2}(\frac{2}{n}+1)\sigma^2 + (\frac{1}{n}+8\chi+1)\zeta^2)\,,$$

- *(**non-convexity**) if each function is $L$-smooth, there are some constants $a, b > 0$ such that*

$$\frac{1}{T}\int_0^{\mathsf{T}} \mathbb{E}\|\nabla f(x_t)\|^2\, dt \leq \frac{4aL(1+\chi)}{T}(f(x_0) - f(x^*)) + b\Big(\sqrt{\frac{L(f(x_0)-f(x^*))}{T}((1+\frac{1}{n})\sigma^2 + (1+\frac{1}{n}+\chi)\xi^2)}\Big)\,.$$

*Also, the expected number of gradient step is $nT$ and the number of communications is $\mathrm{Tr}\,(\Lambda)T$.*

| Method | **Strongly Convex** | **Non-Convex** |
|---|---|---|
| Koloskova et al. [2020] | $\frac{\sigma^2}{n\mu^2\epsilon} + \sqrt{L}\frac{\chi_1\xi+\sqrt{\chi_1}\sigma}{\mu^2\sqrt{\epsilon}} + \frac{L}{\mu}\chi_1$ | $\frac{L\sigma^2}{n\epsilon^2} + L\frac{\chi_1\xi+\sqrt{\chi_1}\sigma}{\epsilon^{3/2}} + \frac{L\chi_1}{\epsilon}$ |
| Ours | $\frac{\sigma^2+\sqrt{\chi_1\chi_2}\xi^2}{\mu^2\epsilon} + \frac{L}{\mu}\sqrt{\chi_1\chi_2}$ | $L\frac{\sigma^2+\sqrt{\chi_1\chi_2}\xi^2}{\epsilon^2} + \frac{L\sqrt{\chi_1\chi_2}}{\epsilon}$ |
| Lian et al. [2018] | - | $L\frac{\sigma^2+\xi^2}{\epsilon^2} + \frac{nL\chi_1}{\epsilon}$ |

Table 5.4: Comparison of convergence rates for strongly convex and non-convex objectives against concurrent works in the fixed topology setting. We neglect logarithmic terms. Observe that thanks to the maximal resistance $\chi_2 \leq \chi_1$, our method obtains substantial acceleration for the bias term.

**Implementation.** As observed by a couple of students, implementing an efficient Poisson Process simulator poses significant challenges. A straightforward implementation fails to fully harness the method's maximal speed. This limitation is particularly due to the synchronous nature of the CUDA event manager, which schedules events in a predetermined ordern which is suboptimal for our use case. Therefore, we have adapted our simulator to mitigate these speed limitations. The achieved speed-ups across various datasets are summarized in Table 5.3, where we varied the number of workers from $n = 4$ to $n = 64$.

**Results.** Figure 5.2 illustrates the acceleration gains achievable using $\mathbf{A}^2\mathbf{CiD}^2$. As the number of workers increases, the equivalent batchsize increases [Goyal et al., 2017], which leads to a performance drop, which is particularly visible when scaling up to 64 workers. Although we employed the strategy suggested in Goyal et al. [2017], a performance gap remains. However, our observations indicate that $\mathbf{A}^2\mathbf{CiD}^2$not only minimizes this gap but also consistently enhances performance across various settings. In particular, it also allows to reduce the consensus distance between workers, as predicted by our theory.

**Open Questions.** Variance reduction techniques are generally not applicable to Deep Learning, as indicated in Gower et al. [2020]. However, these techniques aim to reduce the number of gradient calls, and thus a hybrid primal formulation could be helpful, for doing so and remains an open question.

# Chapter 6

# Conclusion and perspectives

## 6.1 Conclusion

Throughout this manuscript, I have proposed an exploration of various research avenues that intertwine graphs and optimization. Chapter 3 presents multiple negative outcomes from my efforts to implement Harmonic Analysis and Signal Processing techniques in the realm of Geometric Deep Learning. Chapters 4 and 5 address methods aimed at enhancing the parallelization potential when training a Deep Neural Network, all while ensuring optimal performance. Specifically, Chapter 4 delves into the back-propagation algorithm, emphasizing model parallelism. In contrast, Chapter 5 offers a systematic exploration of decentralized asynchronous algorithms. Both chapters suggest techniques to further distribute the computational tasks of training. Yet, there remain several challenges that may limit their widespread adoption of the techniques that I presented, notably:

- **Lack of practical implementations.** Many of the algorithms I have introduced are based on simulators or pseudo-simulators rather than practical implementations, which reduce their impact and benefits.
- **Limits of asynchronous/randomized decentralized methods.** While the deterministic decentralized framework appears well-understood [Scaman et al., 2017], the challenge of defining and achieving the lower bounds for asynchronous methods remains an open question. This aims at filling Table 5.1.
- **Beyond supervised models and vision tasks.** The majority of the methodologies I have discussed are tuned for a supervised vision model, which is a significant restriction.
- **Identifying the inductive biases of Geometric Deep Learning.** The manipulation of many objects would be of interest for geometric Deep

Learning (for instance, vector fields, Gauge, sheaf...), which would allow to design differentiable models amenable to process supervise those data.

- **Reducing accuracy gap.** Surely, the most challenging lock is to find techniques to reduce the current accuracy gap with End-to-End back-propagation training.

## 6.2 Perspectives and Future Contributions

Looking ahead, I am deeply committed to further expanding my research in distributed learning and the interactions with other discplines. A central long-term goal of mine is to design a collaborative framework for training gigantic models tailored to scientific applications. This approach would leverage collaborative resources, echoing initiatives seen in other domains, such as Anderson et al. [2002], where users on the internet collaborate to annotate astronomical datasets. With this in mind, I would like to outline several research directions that I am eager to delve into, in the coming years:

**Efficient Integration of Data-Parallelism and Model-Parallelism.** Chapters 4 and 5 delve into model-parallelism and data-parallelism, respectively. These approaches segment layers and data into smaller portions for parallel processing. While each two of those training methodology holds potential for being independently combined, offering enhanced levels of parallelization, the advantages and trade-offs of merging them are not always evident and there should be some automatic algorithms to decide an optimal type of parallelism. In fact, questions linger regarding the computational and performance optimality of such combinations, depending on the hardware. Furthermore, I would like to design a unified solution, using both model and data parallelism, that would improve upon the standard but independent combination of those two paradigms.

For example, one scenario which interest me is to train large swarm of layers, i.e., Neural Networks whose multiple copies of layers are independently processed yet which should behave as a single model. Here, I am curious to explore the potential benefits of integrating Local SGD Stich [2018] with Local Learning, such as the reduction in required communication compared to a centralized approach while maintaining a high level of parallelization. My future research endeavors will focus on the formulation of this type of strategy, in order to derive algorithms and methodologies to amplify parallelization and locality.

**Investigating Alternatives to Greedy Learning.** Although Greedy Learning, detailed in Chapter 4, offers distinct advantages, it is primarily suited to supervised contexts, and in particular classification. This is attributed to the necessity of having a preconceived notion of the end task when designing a local objective. Indeed, it is well-known that traditional Deep Neural Networks progressively achieve linear separability [Zeiler and Fergus, 2014]. However,

in generative contexts, for example, integration of multiple image scales is essential [Angles and Mallat, 2018] and there is no clear notion of progressive linear separability. In other words, producing a high-quality image requires an interplay of different scales, necessitating a refinement across these scales. However, for a Local Learning procedure, a layer can only process one scale at time. Such a requirement clashes with the inherent locality of Greedy Learning strategies: indeed, auxiliary classifiers often use small receptive fields (spatially local, without global interaction) or aggressive spatial averagings (which break spatial localization, and thus dilute the notion of scale). U-net [Zhang et al., 2018] could be a potential solution to overcome this issue, however, this makes the solution architecture dependent, and thus less generic.

To make headway towards more generic decentralization and more distributed algorithms, we thus must seek alternatives to Local Greedy Learning. An ideal alternative would ensure that updates rely on localized information while preserving pertinent layer-specific data without dilution from other layers' computations. The notion of stale gradients [Zhuang et al., 2021b] is a potential contender. Being loss-agnostic, it paves the way for more parallelization and it has demonstrated remarkable performance on standard benchmarks, as seen in Xu et al. [2020]. This approach computes gradients in parallel, by storing intermediary activations, resulting in updates governed by delayed gradients. However, challenges like memory overhead, attributable to activation storage, limit the broader adoption of such techniques. I would like to propose alternatives to limit these overhead and to allow this strategy to be competitive, for instance using compression or by unlocking the need to store activations.

**Mitigating the Need for Grid Search.** The promise of decentralized asynchronous learning, as highlighted by Nabli et al. [2023] in Chapter 5, is undeniably compelling, due to the achieved reduction in communication consensus. However, its practical realization has necessitated an extensive benchmarking, which would add on top of the standard ad-hoc engineering trial and error of Neural Network design. The later is primarily done to identify hyper-parameters that avoid potential performance drops. This added computational overhead is highly undesirable, especially given the frequent need to recalibrate hyper-parameters when using a new datasets or new model. Note that for established datasets and standard baselines, this procedure is often non-existent, as the community consistently refines the "optimal" hyper-parameters, ensuring top performance [Paszke et al., 2019]. However, such collaborative optimization has not yet been deployed in the realm of decentralized learning.

One of my objectives would be to propose a methodology to produce robust and generic starting hyper-parameters for decentralized learning. However, given the lack of predictive theoretical results in decentralized Deep Learning, this pursuit is challenging since current practices are predominantly empirical. A promising direction seems to consider meta-learning tools, such as [Metz et al., 2022]. These tools, despite tailored for centralized setups, train small Convolutional Neural Networks across a variety of commonly encountered problems in

application, faster than their hand-engineered counter-part. The basic idea is to learn an optimizer to train faster a collection of models. In a decentralized setting, such optimizers would have to be topology aware, while being local and aiming at reducing communication, which is a significant difference with this setting. Such tools could substantially reduce reliance on exhaustive grid searches, setting the stage for less grid-search dependent fine-tuning of decentralized frameworks in the future.

**Collaborative Training with Guarantees.** One ambitious goal of my research, which I aspire to realize in the coming years, is to craft a novel collaborative training framework. Its goal would be to facilitate secure, efficient, and dependable collaborative training across diverse organizations or computational nodes and clusters. At present, ultra-scale computing remains exclusive to a minority of people with access to super-computers and clusters. However, initiatives such as Anderson et al. [2002], which champion resource-sharing among users, have catalyzed remarkable advancements in various fields (here, in astronomy). Delving into asynchronous decentralized techniques and exploring how to segment colossal models into more manageable fragments can pave the way to scalability in such endeavors.

Thus an objective of my research is to introduce models and algorithms that empower open-source platforms to agregate together a vast array of resources. While initiatives like [Ryabinin and Gusev, 2020, team, 2020, Rao et al., 2020, Diskin et al., 2021, Yuan et al., 2022, Borzunov et al., 2022] already exist, I am convinced that an additional layer of parallelization and decentralization could further improve what is currently available. Not only should those methods be validated empirically in practical environments and, optimistically, yet they should also lead to the development of valuable tools for the research community, such as open-source libraries and softwares.

**Machine Learning Models for Scientific Applications:** One other of my foremost objective is to harness the capabilities of large models and learning algorithms for the advancement of scientific research. Indeed, Deep Learning models have immense potential to push the boundaries of other disciplines [Rish, 2023, Taylor et al., 2022]. My aspiration is to reposition Deep Learning not just as a supplementary engineering tool, but as an integral driver of innovation across fields like biology, physics, mathematics, chemistry, and beyond. I posit that Deep Learning can lead to novel ideas for researchers and offer groundbreaking directions, thus serving as a catalyst in these scientific disciplines.

In the future, aligned with my endeavors in collaborative training, I believe in the training of large-scale models (e.g., LLMs, foundation models...) tailored for scientific data or even academic papers. By fine-tuning those models trained on specific tasks that researchers try to solve, my aim is to enhance the processing of voluminous scientific productions. This has already significantly sped up scientific discoveries by reducing the time needed for data analytics, evaluation, simulations and inferential reasoning [Reichstein et al., 2019, Choudhary et al.,

2022, Bhatt et al., 2021, Lan et al., 2022, Bourilkov, 2019, Guest et al., 2018, Angermueller et al., 2016, Silver et al., 2016, Goh et al., 2017].

Might we develop generic solvers and assistants for research scientists?

# Bibliography


Emmanuel Abbe. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

Alekh Agarwal, Martin J Wainwright, and John C Duchi. Distributed dual averaging in networks. *Advances in Neural Information Processing Systems*, 23, 2010.

Vicki H Allan, Reese B Jones, Randall M Lee, and Stephen J Allan. Software pipelining. *ACM Computing Surveys (CSUR)*, 27(3):367–432, 1995.

David P Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@ home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

Mathieu Andreux, Tomás Angles, Georgios Exarchakisgeo, Robertozzi Leonardu, Gaspar Rochette, Louis Thiry, John Zarka, Stéphane Mallat, Joakim Andén, Eugene Belilovsky, et al. Kymatio: Scattering transforms in python. *The Journal of Machine Learning Research*, 21(1):2256–2261, 2020.

Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular systems biology*, 12(7):878, 2016.

Tomás Angles and Stéphane Mallat. Generative networks as inverse problems with scattering transforms. In *International Conference on Learning Representations*, 2018.

Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *International Conference on Machine Learning*, pages 584–592, 2014.

Mahmoud Assran, Nicolas Loizou, Nicolas Ballas, and Mike Rabbat. Stochastic gradient push for distributed deep learning. In *International Conference on Machine Learning*, pages 344–353. PMLR, 2019.

Medha Atre, Birendra Jha, and Ashwini Rao. Distributed deep learning using volunteer computing-like paradigm. In *2021 IEEE International Parallel*

*and Distributed Processing Symposium Workshops (IPDPSW)*, pages 933–942. IEEE, 2021.

Kenneth Atz, Francesca Grisoni, and Gisbert Schneider. Geometric deep learning on molecular representations. *Nature Machine Intelligence*, 3(12):1023–1032, 2021.

Henri E Bal, Jennifer G Steiner, and Andrew S Tanenbaum. Programming languages for distributed computing systems. *ACM Computing Surveys (CSUR)*, 21(3):261–322, 1989.

Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35: 21750–21764, 2022.

Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in neural information processing systems*, 31, 2018.

Atılım Güneş Baydin, Barak A Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation. *arXiv preprint arXiv:2202.08587*, 2022.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pages 583–593. PMLR, 2019.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. In *International Conference on Machine Learning*, pages 736–745. PMLR, 2020.

Eugene Belilovsky, Louis Leconte, Lucas Caccia, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns for synchronous and asynchronous distributed learning. *arXiv preprint arXiv:2106.06401*, 2021.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 2006.

Dimitri Bertsekas and John Tsitsiklis. *Parallel and distributed computation: numerical methods*. Athena Scientific, 2015.

Chandradeep Bhatt, Indrajeet Kumar, V Vijayakumar, Kamred Udham Singh, and Abhishek Kumar. The state of the art of deep learning models in medical science and their challenges. *Multimedia Systems*, 27(4):599–613, 2021.

Michael Blot, David Picard, Matthieu Cord, and Nicolas Thome. Gossip training for deep learning. In *Advances in Neural Information Processing Systems*, volume 30, 2016.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Chamberlain, Pietro Liò, and Michael Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *Advances in Neural Information Processing Systems*, 35:18527–18541, 2022.

Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. Petals: Collaborative inference and fine-tuning of large models. *arXiv preprint arXiv:2209.01188*, 2022.

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20, 2007.

Dimitri Bourilkov. Machine and deep learning applications in particle physics. *International Journal of Modern Physics A*, 34(35):1930019, 2019.

Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Advanced lectures on machine learning*, pages 169–207. Springer, 2004.

S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006a. doi: 10.1109/TIT.2006.874516.

Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6): 2508–2530, 2006b.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1): 1–122, 2011.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Joan Bruna. *Scattering representations for recognition*. PhD thesis, Ecole Polytechnique X, 2013.

Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

Roy Harold Campbell, Thomas Anderson, and Brian Randell. Practical fault tolerant software for asynchronous systems. In *Safety of Computer Control Systems 1983 (Safecomp'83)*, pages 59–65. Elsevier, 1983.

Ashok K Chandra, Prabhakar Raghavan, Walter L Ruzzo, Roman Smolensky, and Prasoon Tiwari. The electrical resistance of a graph captures its commute and cover times. *computational complexity*, 6(4):312–340, 1996.

Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

Jack Choquette. Nvidia hopper gpu: Scaling performance. In *2022 IEEE Hot Chips 34 Symposium (HCS)*, pages 1–46. IEEE Computer Society, 2022.

Jack Choquette. Nvidia hopper h100 gpu: Scaling performance. *IEEE Micro*, 2023.

Anna Choromanska, Benjamin Cowen, Sadhana Kumaravel, Ronny Luss, Mattia Rigotti, Irina Rish, Paolo Diachille, Viatcheslav Gurev, Brian Kingsbury, Ravi Tejwani, et al. Beyond backprop: Online alternating minimization with auxiliary variables. In *International Conference on Machine Learning*, pages 1193–1202. PMLR, 2019.

Kamal Choudhary, Brian DeCost, Chi Chen, Anubhav Jain, Francesca Tavazza, Ryan Cohn, Cheol Woo Park, Alok Choudhary, Ankit Agrawal, Simon JL Billinge, et al. Recent advances and applications of deep learning methods in materials science. *npj Computational Materials*, 8(1):59, 2022.

Vaclav Cizek. Discrete hilbert transform. *IEEE Transactions on Audio and Electroacoustics*, 18(4):340–343, 1970.

Taco S Cohen and Max Welling. Steerable cnns. In *International Conference on Learning Representations*, 2016.

Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. In *International Conference on Learning Representations*, 2018.

Ronald R Coifman, Stephane Lafon, Ann B Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven W Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the national academy of sciences*, 102(21):7426–7431, 2005.

Jeff Daily, Abhinav Vishnu, Charles Siegel, Thomas Warfel, and Vinay Amatya. Gossipgrad: Scalable deep learning using gossip communication based asynchronous gradient descent, 2018.

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

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

Michael Diskin, Alexey Bukhtiyarov, Max Ryabinin, Lucile Saulnier, Anton Sinitsin, Dmitry Popov, Dmitry V Pyrkin, Maxim Kashirin, Alexander Borzunov, Albert Villanova del Moral, et al. Distributed deep learning in open collaborations. *Advances in Neural Information Processing Systems*, 34: 7879–7897, 2021.

Michael Eickenberg, Erwan Allys, Azadeh Moradinezhad Dizgah, Pablo Lemos, Elena Massara, Muntazir Abidi, ChangHoon Hahn, Sultan Hassan, Bruno Regaldo-Saint Blancard, Shirley Ho, et al. Wavelet moments for cosmological parameter estimation. *arXiv preprint arXiv:2204.07646*, 2022.

Tolga Ergen and Mert Pilanci. Implicit convex regularizers of cnn architectures: Convex optimization of two-and three-layer networks in polynomial time. In *International Conference on Learning Representations (ICLR) 2021*, 2021.

Mathieu Even, Raphaël Berthier, Francis Bach, Nicolas Flammarion, Hadrien Hendrikx, Pierre Gaillard, Laurent Massoulié, and Adrien Taylor. Continuized accelerations of deterministic and stochastic gradient descents, and of gossip algorithms. *Advances in Neural Information Processing Systems*, 34: 28054–28066, 2021.

Louis Fournier, Stéphane Rivaud, Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Can forward gradient match backpropagation? In *Fortieth International Conference on Machine Learning*, 2023.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, pages 1165–1173. PMLR, 2017.

James Fung and Steve Mann. Openvidia: parallel gpu computer vision. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 849–852, 2005.

Fernando Gama, Alejandro Ribeiro, and Joan Bruna. Stability of graph scattering transforms. In *Advances in Neural Information Processing Systems*, pages 8036–8046, 2019.

Charles Godfrey, Davis Brown, Tegan Emerson, and Henry Kvinge. On the symmetries of deep learning models and their internal representations. *Advances in Neural Information Processing Systems*, 35:11893–11905, 2022.

Garrett B Goh, Nathan O Hodas, and Abhinav Vishnu. Deep learning for computational chemistry. *Journal of computational chemistry*, 38(16):1291–1307, 2017.

Robert M Gower, Mark Schmidt, Francis Bach, and Peter Richtárik. Variance-reduced methods for machine learning. *Proceedings of the IEEE*, 108(11): 1968–1983, 2020.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Klaus Greff, Rupesh K Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. *arXiv preprint arXiv:1612.07771*, 2016.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35:507–520, 2022.

Nathan Grinsztajn, Louis Leconte, Philippe Preux, and Edouard Oyallon. Interferometric graph transform for community labeling. *arXiv preprint arXiv:2106.05875*, 2021a.

Nathan Grinsztajn, Philippe Preux, and Edouard Oyallon. Low-rank projections of gcns laplacian. *ICLR Workshop GTRL*, 2021b.

Dan Guest, Kyle Cranmer, and Daniel Whiteson. Deep learning and its application to lhc physics. *Annual Review of Nuclear and Particle Science*, 68: 161–181, 2018.

David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

Donghyeon Han, Gwangtae Park, Junha Ryu, and Hoi-jun Yoo. Extension of direct feedback alignment to convolutional and recurrent neural network for bio-plausible deep learning. *arXiv preprint arXiv:2006.12830*, 2020.

Han Han and Vincent Lostanlen. Wav2shape: Hearing the shape of a drum machine. In *Forum Acusticum*, pages 647–654, 2020.

XY Han, Vardan Papyan, and David L Donoho. Neural collapse under mse loss: Proximity to and dynamics on the central path. In *International Conference on Learning Representations*, 2021.

Hadrien Hendrikx. A principled framework for the design and analysis of token algorithms, 2022.

Yu-Guan Hsieh, Yassine Laguel, Franck Iutzeler, and Jérôme Malick. Push–pull with device sampling. *IEEE Transactions on Automatic Control*, 2023.

Furong Huang, Jordan Ash, John Langford, and Robert Schapire. Learning deep resnet blocks sequentially using boosting theory. In *International Conference on Machine Learning*, pages 2058–2067. PMLR, 2018.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

Travis S Humble, Alexander McCaskey, Dmitry I Lyakh, Meenambika Gowrishankar, Albert Frisch, and Thomas Monz. Quantum computers for high-performance computing. *IEEE Micro*, 41(5):15–23, 2021.

Zhouyuan Huo, Bin Gu, and Heng Huang. Training neural networks using features replay. *Advances in Neural Information Processing Systems*, 2018a.

Zhouyuan Huo, Bin Gu, and Heng Huang. Training neural networks using features replay. *Advances in Neural Information Processing Systems*, 31, 2018b.

Zhouyuan Huo, Bin Gu, qian Yang, and Heng Huang. Decoupled parallel backpropagation with convergence guarantee. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2098–2106, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018c. PMLR. URL `http://proceedings.mlr.press/v80/huo18a.html`.

A. G. Ivakhnenko and V.G. Lapa. Cybernetic predicting devices. *CCM Information Corporation*, 1965.

Jörn-Henrik Jacobsen, Arnold WM Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. In *International Conference on Learning Representations*, 2018.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

M Jaderberg, K Simonyan, A Vedaldi, and A Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. In *NIPS Deep Learning Workshop*. Neural Information Processing Systems, 2014.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. *International Conference of Machine Learning*, 2017a.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *International conference on machine learning*, pages 1627–1635. PMLR, 2017b.

Michael Kampffmeyer, Yinbo Chen, Xiaodan Liang, Hao Wang, Yujia Zhang, and Eric P Xing. Rethinking knowledge graph propagation for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11487–11496, 2019.

Ioannis Karatzas and Steven Shreve. *Brownian motion and stochastic calculus*, volume 113. Springer Science & Business Media, 1991.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

Douglas J Klein and Milan Randić. Resistance distance. *Journal of mathematical chemistry*, 12(1):81–95, 1993.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pages 5381–5393. PMLR, 2020.

Anastasiia Koloskova, Tao Lin, and Sebastian U Stich. An improved analysis of gradient tracking for decentralized machine learning. *Advances in Neural Information Processing Systems*, 34:11422–11435, 2021.

Dmitry Kovalev, Adil Salim, and Peter Richtarik. Optimal and practical algorithms for smooth and strongly convex decentralized optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18342–18352. Curran Associates, Inc., 2020.

Dmitry Kovalev, Elnur Gasanov, Alexander Gasnikov, and Peter Richtárik. Lower bounds and optimal algorithms for smooth and strongly convex decentralized optimization over time-varying networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021a.

Dmitry Kovalev, Egor Shulgin, Peter Richtárik, Alexander V Rogozin, and Alexander Gasnikov. Adom: accelerated decentralized optimization method for time-varying networks. In *International Conference on Machine Learning*, pages 5784–5793. PMLR, 2021b.

Dmitry Kovalev, Alexander Gasnikov, and Peter Richtárik. Accelerated primal-dual gradient method for smooth and convex-concave saddle-point problems with bilinear coupling. *Advances in Neural Information Processing Systems*, 35:21725–21737, 2022.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Griffin Lacey, Graham W Taylor, and Shawki Areibi. Deep learning on fpgas: Past, present, and future. *arXiv preprint arXiv:1602.04283*, 2016.

Zhiquan Lai, Shengwei Li, Xudong Tang, Keshi Ge, Weijie Liu, Yabo Duan, Linbo Qiao, and Dongsheng Li. Merak: An efficient distributed dnn training framework with automated 3d parallelism for giant foundation models. *IEEE Transactions on Parallel and Distributed Systems*, 34(5):1466–1478, 2023.

Yihuai Lan, Lei Wang, Qiyuan Zhang, Yunshi Lan, Bing Tian Dai, Yan Wang, Dongxiang Zhang, and Ee-Peng Lim. Mwptoolkit: an open-source framework for deep learning-based math word problem solvers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 13188–13190, 2022.

Julien Launay, Iacopo Poli, François Boniface, and Florent Krzakala. Direct feedback alignment scales to modern deep learning tasks and architectures. *Advances in neural information processing systems*, 33:9346–9360, 2020.

Quoc V Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE, 2013.

Rémi Leblond, Fabian Pedregosa, and Simon Lacoste-Julien. Asaga: Asynchronous parallel saga. In *20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017*, 2017.

Yann LeCun. 1.1 deep learning hardware: Past, present, and future. In *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 12–19. IEEE, 2019.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I 15*, pages 498–515. Springer, 2015.

Gwen Legate, Nicolas Bernier, Lucas Caccia, Edouard Oyallon, and Eugene Belilovsky. Guiding the last layer in federated learning with pre-trained models. *Advances in Neural Information Processing Systems*, 2023.

Ang Li, Shuaiwen Leon Song, Jieyang Chen, Jiajia Li, Xu Liu, Nathan R Tallent, and Kevin J Barker. Evaluating modern gpu interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect. *IEEE Transactions on Parallel and Distributed Systems*, 31(1):94–110, 2019a.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Shenggui Li, Fuzhao Xue, Yongbin Li, and Yang You. Sequence parallelism: Making 4d parallelism possible. *arXiv preprint arXiv:2105.13120*, 2021.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020. doi: 10.1109/MSP.2020.2975749.

Zhi Li, Wei Shi, and Ming Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *IEEE Transactions on Signal Processing*, 67(17):4494–4506, 2019b.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in neural information processing systems*, 30, 2017.

Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning*, pages 3043–3052. PMLR, 2018.

Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random feedback weights support learning in deep neural networks. *CoRR*, abs/1411.0247, 2014a. URL `http://arxiv.org/abs/1411.0247`.

Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*, 2014b.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Thorben Louw and Simon McIntosh-Smith. Using the graphcore ipu for traditional hpc applications. In *3rd Workshop on Accelerated Machine Learning (AccML)*, 2021.

Eran Malach and Shai Shalev-Shwartz. A provably correct algorithm for deep learning that actually works. *arXiv preprint arXiv:1803.09522*, 2018.

Eran Malach and Shai Shalev-Shwartz. Is deeper better only when shallow is good? *Advances in Neural Information Processing Systems*, 32, 2019.

Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

Stéphane Mallat. Recursive interferometric representation. In *Proc. of EUSICO conference, Danemark*, 2010.

Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203, 2016.

Dmitry Metelev, Alexander Rogozin, Dmitry Kovalev, and Alexander Gasnikov. Is consensus acceleration possible in decentralized optimization over slowly time-varying networks? In *International Conference on Machine Learning*, pages 24532–24554. PMLR, 2023.

Luke Metz, James Harrison, C Daniel Freeman, Amil Merchant, Lucas Beyer, James Bradbury, Naman Agrawal, Ben Poole, Igor Mordatch, Adam Roberts, et al. Velo: Training versatile learned optimizers by scaling up. *arXiv preprint arXiv:2211.09760*, 2022.

Jonathan Milton and Payman Zarkesh-Ha. Impacts of topology and bandwidth on distributed shared memory systems. *Computers*, 12(4):86, 2023.

Konstantin Mishchenko, Francis Bach, Mathieu Even, and Blake E Woodworth. Asynchronous sgd beats minibatch sgd under arbitrary delays. *Advances in Neural Information Processing Systems*, 35:420–433, 2022.

Sparsh Mittal and Jeffrey S Vetter. A survey of cpu-gpu heterogeneous computing techniques. *ACM Computing Surveys (CSUR)*, 47(4):1–35, 2015.

Hesham Mostafa, Vishwajith Ramesh, and Gert Cauwenberghs. Deep supervised learning using local errors. *Frontiers in neuroscience*, 12:608, 2018.

Adel Nabli and Edouard Oyallon. Decentralized asynchronous optimization with dadao allows decoupling and acceleration. *preprint*, 2023a.

Adel Nabli and Edouard Oyallon. Dadao: Decoupled accelerated decentralized asynchronous optimization. In *International Conference on Machine Learning*, pages 25604–25626. PMLR, 2023b.

Adel Nabli, Eugene Belilovsky, and Edouard Oyallon. $\mathbf{A^2CiD^2}$: Accelerating asynchronous communication in decentralized deep learning. *Advances in Neural Information Processing Systems*, 2023.

Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1): 48–61, 2009.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29, 2016a.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29, 2016b.

Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. In *International conference on machine learning*, pages 4839–4850. PMLR, 2019.

Hoang NT and Takanori Maehara. Frequency analysis for graph convolution network, 2020. URL `https://openreview.net/forum?id=HylthC4twr`.

Alexander G Ororbia and Ankur Mali. Biologically motivated algorithms for propagating local target representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4651–4658, 2019.

Alexander G Ororbia, Ankur Mali, Daniel Kifer, and C Lee Giles. Conducting credit assignment by aligning local representations. *arXiv preprint arXiv:1803.01834*, 2018.

Edouard Oyallon. A hybrid network: Scattering and convnet. *Submission on OpenReview*, 2016.

Edouard Oyallon. *Analyzing and introducing structures in deep convolutional neural networks*. PhD thesis, Paris Sciences et Lettres (ComUE), 2017a.

Edouard Oyallon. Building a regular decision boundary with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5106–5114, 2017b.

Edouard Oyallon. Interferometric graph transform: a deep unsupervised graph representation. In *International Conference on Machine Learning*, pages 7434–7444. PMLR, 2020.

Edouard Oyallon. Advanced topics in deep learning. *Lectures' draft at IPP*, 2021.

Edouard Oyallon and Stéphane Mallat. Deep roto-translation scattering for object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2865–2873, 2015.

Edouard Oyallon and Julien Rabin. An Analysis of the SURF Method. *Image Processing On Line*, 5:176–218, 2015. `https://doi.org/10.5201/ipol.2015.69`.

Edouard Oyallon, Stéphane Mallat, and Laurent Sifre. Generic deep networks with wavelet scattering. *arXiv preprint arXiv:1312.5940*, 2013.

Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5618–5627, 2017.

Edouard Oyallon, Sergey Zagoruyko, Gabriel Huang, Nikos Komodakis, Simon Lacoste-Julien, Matthew Blaschko, and Eugene Belilovsky. Scattering networks for hybrid representation learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2208–2221, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R So, Maud Texier, and Jeff Dean. The carbon footprint of machine learning training will plateau, then shrink. *Computer*, 55(7):18–28, 2022.

Yuma Rao, Jacob Steeves, Ala Shaabana, Daniel Attevelt, and Matthew McAteer. Bittensor: A peer-to-peer intelligence market. *arXiv preprint arXiv:2003.03917*, 2020.

Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, and fnm Prabhat. Deep learning and process understanding for data-driven earth system science. *Nature*, 566(7743):195–204, 2019.

Mengye Ren, Simon Kornblith, Renjie Liao, and Geoffrey Hinton. Scaling forward gradient with local losses. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=JxpBP1JM15-`.

Irina Rish. Incite project: Scalable foundation models for transferrable generalist ai, 2023. URL `https://sites.google.com/view/irinarish/incite-project?authuser=0`. Accessed: 2023-09-01.

Max Ryabinin and Anton Gusev. Towards crowdsourced training of large neural networks using decentralized mixture-of-experts. *Advances in Neural Information Processing Systems*, 33:3659–3672, 2020.

Youcef Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Mathematics of Computation*, 42(166):567–588, 1984.

Adil Salim, Laurent Condat, Dmitry Kovalev, and Peter Richtárik. An optimal algorithm for strongly convex minimization under affine constraints. In *International Conference on Artificial Intelligence and Statistics*, pages 4482–4498. PMLR, 2022.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

Lawrence K Saul and Sam T Roweis. An introduction to locally linear embedding. *unpublished. Available at: http://www. cs. toronto. edu/˜ roweis/lle/publications. html*, 2000.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3027–3036. PMLR, 06–11 Aug 2017.

Grégoire Sergeant-Perthuis, Jakob Maier, Joan Bruna, and Edouard Oyallon. On non-linear operators for geometric deep learning. *Advances in Neural Information Processing Systems*, 35:10984–10995, 2022.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Anirudh Goyal, Ivo Danihelka, Matteo Hessel, and Hado van Hasselt. Learning by directional gradient descent. In *International Conference on Learning Representations*, 2021.

Zhuoqing Song, Lei Shi, Shi Pu, and Ming Yan. Optimal gradient tracking for decentralized optimization. *Mathematical Programming*, pages 1–53, 2023.

Sebastian U Stich. Local sgd converges fast and communicates little. In *International Conference on Learning Representations*, 2018.

Jaspal Subhlok, James M Stichnoth, David R O'hallaron, and Thomas Gross. Exploiting task and data parallelism on a multicomputer. In *Proceedings of the fourth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 13–22, 1993.

Vivienne Sze, Yu-Hsin Chen, Joel Emer, Amr Suleiman, and Zhengdong Zhang. Hardware for machine learning: Challenges and opportunities. In *2017 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–8. IEEE, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable admm approach. In *International conference on machine learning*, pages 2722–2731. PMLR, 2016.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

Learning@home team. Hivemind: a Library for Decentralized Deep Learning. `https://github.com/learning-at-home/hivemind`, 2020.

Irene Tenison, Sai Aravind Sreeramadas, Vaikkunth Mugunthan, Edouard Oyallon, Eugene Belilovsky, and Irina Rish. Gradient masked averaging for federated learning. *arXiv e-prints*, pages arXiv–2201, 2022.

Louis THIRY, Michael Arbel, Eugene Belilovsky, and Edouard Oyallon. The unreasonable effectiveness of patches in deep convolutional kernels methods. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=aYuZO9DIdnn`.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.

Arun Venkitaraman, Saikat Chatterjee, and Peter Händel. On hilbert transform, analytic signal, and modulation analysis for signals over graphs. *Signal Processing*, 156:106–115, 2019.

Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. In *International Conference on Machine Learning*, pages 23341–23362. PMLR, 2022.

Yu Emma Wang, Gu-Yeon Wei, and David Brooks. Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*, 2019.

Blake Woodworth and Nathan Srebro. Lower bound for randomized first order convex optimization. *arXiv preprint arXiv:1709.03594*, 2017.

Blake Woodworth, Kumar Kshitij Patel, Sebastian Stich, Zhen Dai, Brian Bullins, Brendan Mcmahan, Ohad Shamir, and Nathan Srebro. Is local sgd better than minibatch sgd? In *International Conference on Machine Learning*, pages 10334–10343. PMLR, 2020.

Blake E Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

An Xu, Zhouyuan Huo, and Heng Huang. On the acceleration of deep learning model parallelism with staleness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2088–2097, 2020.

Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, 55(1):407–474, 2022.

Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy S Liang, Christopher Re, and Ce Zhang. Decentralized training of foundation models in heterogeneous environments. *Advances in Neural Information Processing Systems*, 35:25464–25477, 2022.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.

Ziming Zhang, Yuting Chen, and Venkatesh Saligrama. Efficient training of very deep neural networks for supervised hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1487–1495, 2016.

Huiping Zhuang, Yi Wang, Qinglai Liu, and Zhiping Lin. Fully decoupled neural network learning using delayed gradients. *IEEE transactions on neural networks and learning systems*, 33(10):6013–6020, 2021a.

Huiping Zhuang, Zhenyu Weng, Fulin Luo, Toh Kar-Ann, Haizhou Li, and Zhiping Lin. Accumulated decoupled learning with gradient staleness mitigation for convolutional neural networks. In *International Conference on Machine Learning*, pages 12935–12944. PMLR, 2021b.