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

pour défendre l'Habilitation à Diriger des Recherches

Edouard Oyallon
CNRS, ISIR, SU

Jury :

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

Part 1: Introduction

Background

Jobs:

Master studies

Ph.D

Postdoc

Assistant Professor

Associate Research Scientist ...

Tenured Research Scientist ...

2012 2014 2016 2018 2020 2022 2024

Teaching activities:

T.A. in probability

Lecturer in DL, RL

Part-time Associate Professor

T.A. in DL

Research topics:

Scattering Transform

Geometric Deep Learning ...

Old school vision

Non Deep baselines to solve Machine Learning problems ...

Interpretability in Deep Learning

Federated Learning ...

Local Learning ...

Decentralized Learning ...

Collaborations/supervisions:

ADONIS

Cosupervision

Dr. Gaël Varoquaux

Léo Grinsztajn

EO

Adel Nabli

Louis Fournier

Dr. Stéphane Rivaud

Collaborators

Dr. Michael Eickenberg

Pr. Eugene Belilovsky

PI of ANR ADONIS (220k€) ...

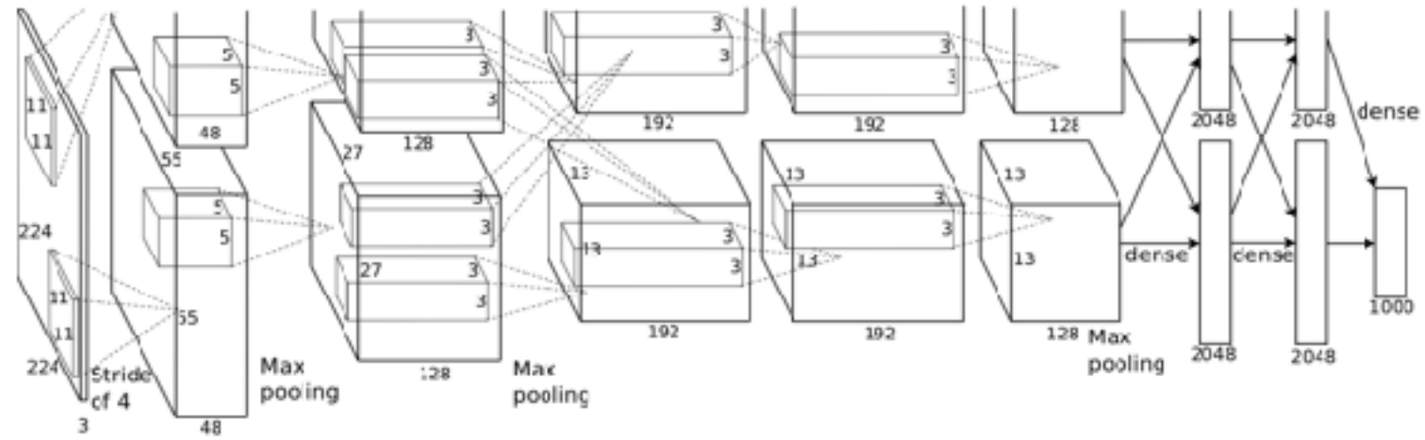
PI of Emergence ADONIS (60k€)

Grants:

co-PI of PEPR SHARP (740k€) ...

Context

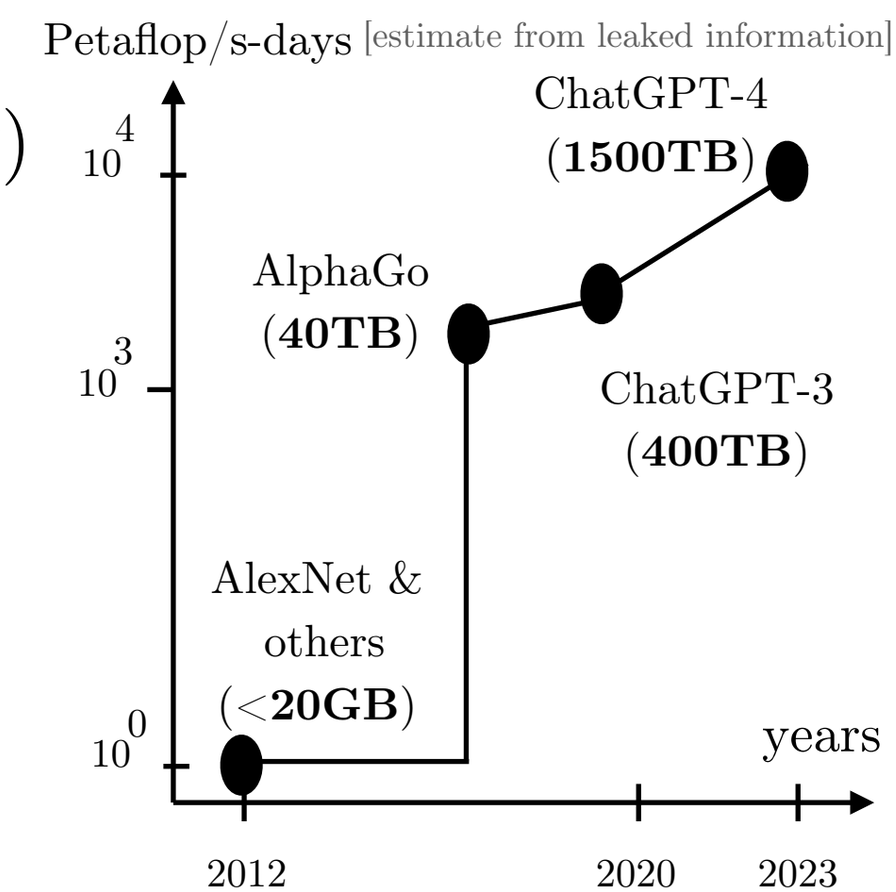
Ref.: ImageNet Classification with Deep Convolutional Neural Networks. A Krizhevsky et al.



Large high-dimensional dataset:

$$\mathcal{D} = \{(x_i, y_i)\}_{i \leq N}$$

- **Deep Learning**: a generic tool to *solve* high-dimensional tasks. One tries to design Φ and to find the best parameter θ so that $\Phi(x, \theta) \approx y$
- **Earlier and current work**: how, why does Deep Learning reach spectacular performance?
- Training foundational models (e.g., ChatGPT) is a data-greedy, computationally intensive and time-consuming task.
- **Current work**: how to maximise resources efficiency still with *scale*? (maximise GPU performance, avoiding buying new infrastructure)

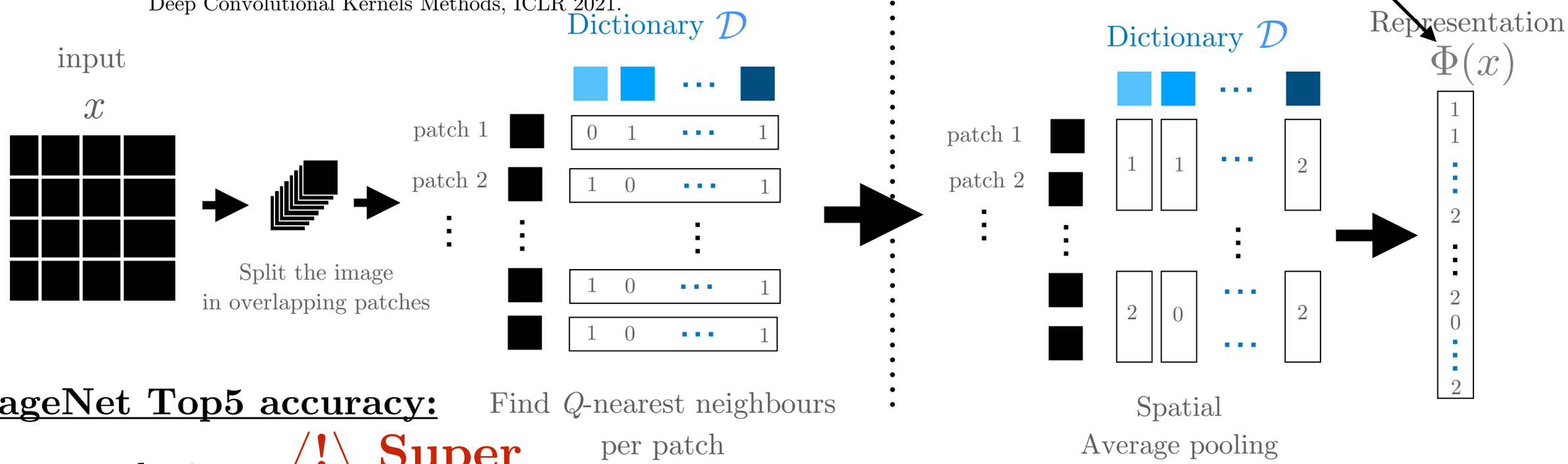


A motivating work

typically 0.5GB per sample

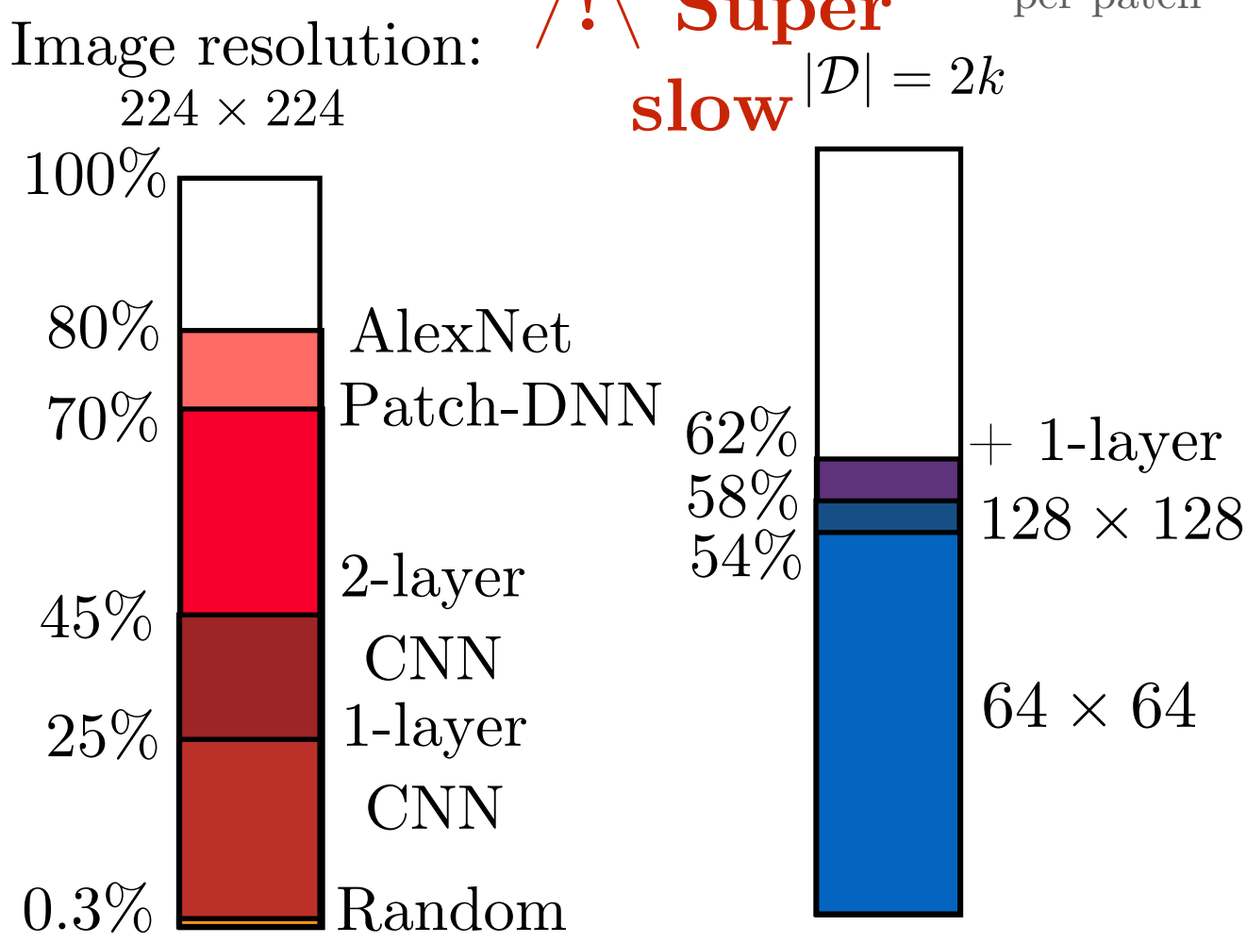
Fix a dictionary of random patches $\mathcal{D}$

Ref.: Thiry L., Arbel M., Belilovsky E. and EO - The Unreasonable Effectiveness of Patches in Deep Convolutional Kernels Methods, ICLR 2021.



ImageNet Top5 accuracy:

/!\ **Super slow**



- "- Is this simply a bag of patches?"
  (Pr. Jean Ponce comment @ Prairie-talks 2021)
- Two major questions:
  - Too simple. Why, how? (**Axis 1**)
  - Too long to train, inefficient. Scalability? (**Axis 2**)

Axis 1: Foundations of Geometric Deep Learning

- Let $\mathcal{M}$ a smooth manifold, and a symmetry group $G \subset \mathrm{Diff}_\infty(\mathcal{M})$. **Example**: translations for $\mathbb{R}^n$, rotations for $\mathcal{S}^{n-1}$.

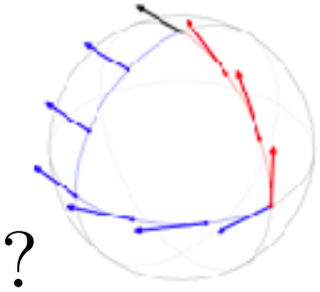
- **The real case**: Consider integrable signals

$$L^2(\mathcal{M}) = \{x : \mathcal{M} \to \mathbb{C}, \int_{\mathcal{M}} |x(u)|^2 \, du < \infty\}$$

and for $\phi \in G$, we define the action L_ϕ on $L^2(\mathcal{M})$ via

$$L_\phi x(u) \triangleq x(\phi^{-1}(u))$$

- **Objective**: designing $\Phi : L^2(\mathcal{M}) \to \mathbb{R}^d$ which allows to provide a useful representation of the data? Covariance: $\Phi L_\phi x = L_\phi \Phi x$ or invariance $\Phi L_\phi x = \Phi x$. **Example**: Convolutions and sum for $\mathbb{R}^n$.

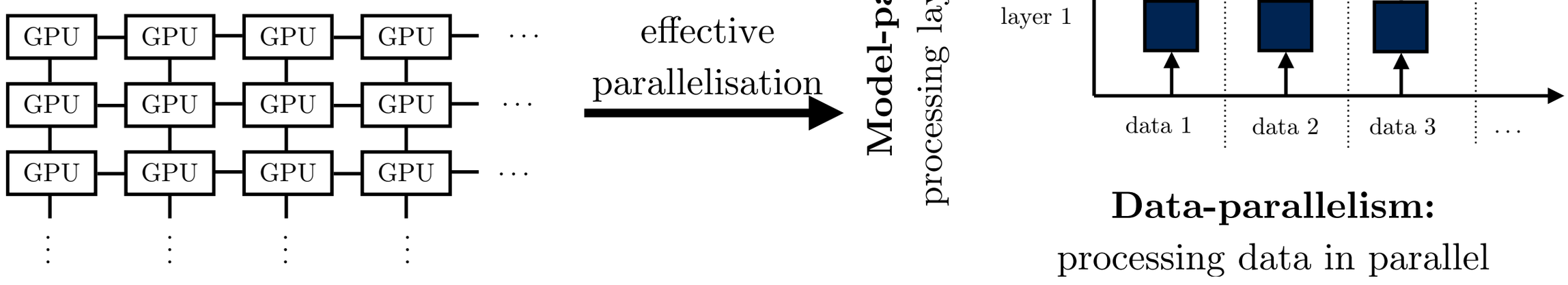


- More simply, is there a *canonical* equivalent of convolutions on manifolds? Are there principled architectures?

Axis 2: More efficient training via Asynchronous and Decentralized Models

- **Parallelisation is key**: Current approach rely on highly connected machines/clusters, mostly centralized and highly synchronous which require expensive, dedicated hardware.

- The main parallelisation techniques are model parallelism and data-parallelism.



- Can we limit the need for those machines/clusters by aggregating distant, multiple resources while speeding-up training, lowering electricity consumption, increasing model size and model accuracy?

- **How?** theoretical guarantees, asynchrony and decentralization.

Part 2: Contributions to this manuscript

Contribution 1: Specifying representation biases for Geometric Deep Learning

Ref.: **[1]** Scattering Representations for Recognition, Phd Thesis, Joan Bruna

Jakob Maier

Pr. Joan Bruna

Pr. Grégoire Sergeant-Perthuis

OE - Interferometric Graph Transform: a Deep Unsupervised Graph Representation, ICML 2020.

- Developing an algorithm to recover Fourier atoms on flat manifolds (e.g, torus).

Sergeant-Perthuis G., Maier J., Bruna J. and OE - On Non-Linear operators for Geometric Deep Learning, NeurIPS 2022.

For $x \in L^2(\mathcal{M})$ and $\tilde{x} \in L^2(\mathcal{M}, T\mathcal{M})$ and $\phi \in \mathrm{Diff}_\infty(\mathcal{M})$ define $L_\phi x(u) \triangleq x(\phi^{-1}(u))$ and $\tilde{L}_\phi \tilde{x}(u) \triangleq d\phi(u).\tilde{x}(\phi^{-1}(u))$

- **Theorem 1** (extension of [1]): Let $M : L^2(\mathcal{M}) \to L^2(\mathcal{M})$ a regular operator, then $ML_\phi = L_\phi M \iff \exists \psi : \mathbb{R} \to \mathbb{R}, Mx(u) = \psi(x(u))$
- **Theorem 2** (vector fields): Let $M : L^2(\mathcal{M}, T\mathcal{M}) \to L^2(\mathcal{M}, T\mathcal{M})$ a regular operator, then $M\tilde{L}_\phi = \tilde{L}_\phi M \iff \exists \lambda \in \mathbb{R}, Mx = \lambda x$

Contribution 2: Decoupled Greedy Learning of Deep Neural Networks


Louis Fournier


Dr. Stéphane Rivaud


Pr. Eugene Belilovsky


Dr. Michael Eickenberg

Belilovsky E., Eickenberg M. and OE - Greedy Layerwise Learning Can Scale to ImageNet, *ICML 2019*.

- Proposing Local criterion to train Deep Neural Networks in a layer-wise, competitive manner.

Belilovsky E., Eickenberg M. and OE - Decoupled Greedy Learning of CNNs, *ICML 2020.*

- Developing Decoupled Local procedures for training Deep Neural Networks layers in parallel.

Fournier L., Rivaud, S., Belilovsky E., Eickenberg M. and OE - Can Forward Gradient Match Backpropagation?, *ICML 2023*.

- Studying more sophisticated Local gradient estimates via Forward Gradient.

Contribution 3: Asynchronous Decentralized algorithms

Adel Nabli

Pr. Eugene Belilovsky

Nabli A. and OE - DADAO: Decoupled Accelerated Decentralized Asynchronous Optimization, *ICML 2023*.

- Proposing fast convex asynchronous, decoupled, decentralized algorithm.

Nabli A. and OE - Decentralized Asynchronous Optimization with DADAO allows Decoupling and Acceleration, *submitted to JMLR*.

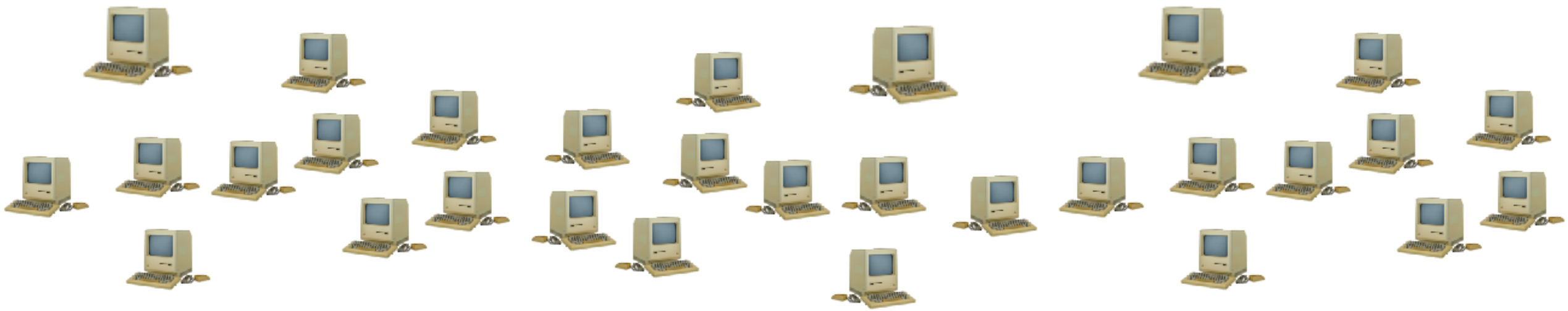
- Refining our knowledge of the graph resistance in the context of distributed optimization.

Nabli A., Belilovsky E. and OE - A²CiD²: Accelerating Asynchronous Communication in Decentralized Deep Learning, *NeurIPS 2023*.

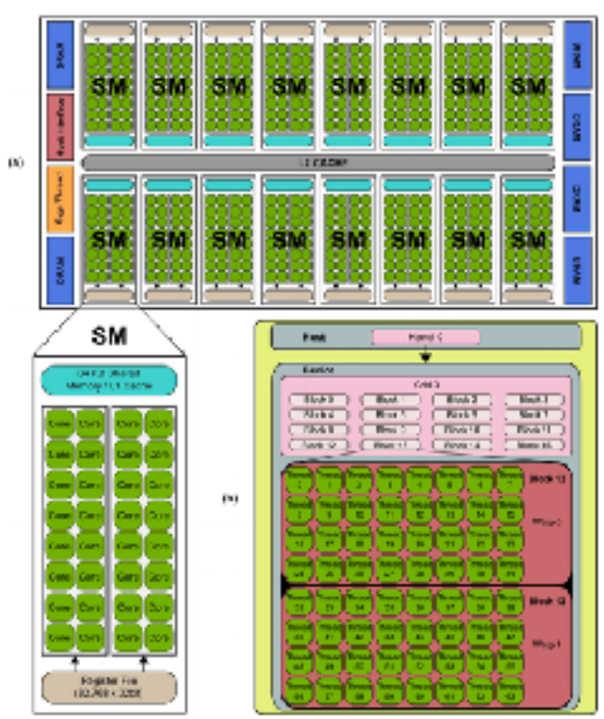
- Reducing the communication rate in the context of decentralized training of Deep Neural Networks.

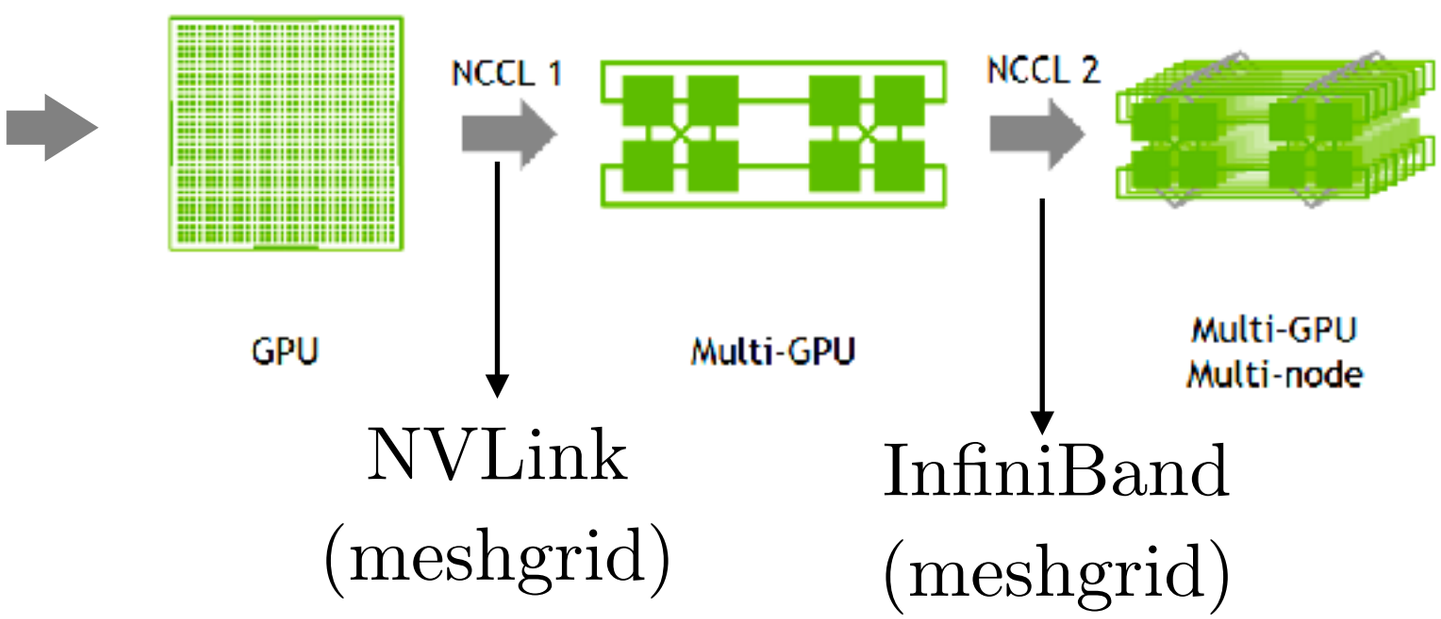
Part 3: Detailed contributions to distributed training of Deep Neural Networks

One word about hardware



- Multiple devices can be aggregated together to collaborate: smartphone, gamer computers, ...
- But this applies too for highly connected clusters, can we benefit minor fluctuations?





Ref.: Nvidia website.

Limits of current approaches for distributed training of Neural Networks

Ref.:Decoupled Neural Interfaces using Synthetic Gradients, Jaderberg et al, 2017

- **Not taking in account model nature**: Backpropagation of Deep Neural Networks has inherent locks (backward, update and forward locks) which impedes speed.

- **Synchronous/simultaneous procedures**: workers wait for each other and execute tasks in a certain order. *(can we break synchrony, and reduce communications/ computations?)*

Objectives: toward a decoupling trick

- **Communications & computations** of decentralized algorithms are typically **coupled**, which results in much undesirable synchronisation and limited potential for distributed computations.

- **Objective 1**: Decoupling gradient computations & communications of back-prop, to allow parallelism, during a backpropagation pass.

- **Objective 2**: Decoupling parameter communications and gradients computations in distributed settings, to maximise hardware efficiency (*perform as much as you can*)

Decoupled Greedy Learning of Deep Neural Networks

(2019-2023)

funded (2021-2022) via ADONIS,

in collaboration with:

Louis Fournier (PhD student)

Dr. Stéphane Rivaud (Postdoc)

Pr. Eugene Belilovsky (Collaborator)

Dr. Michael Eickenberg (Collaborator)

3 publications in:

Asynchronous Decentralized algorithms

(2019-2023)

funded (2021-2022) via ADONIS,

in collaboration with:

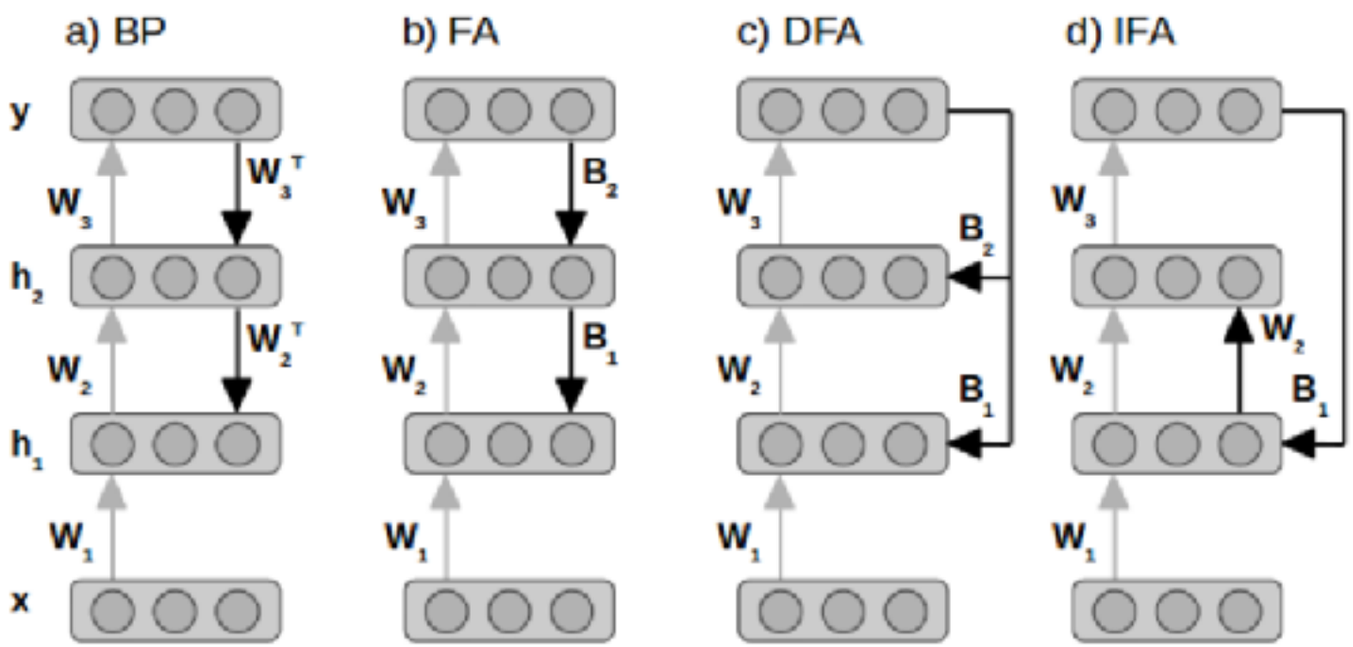
Adel Nabli (PhD student)

Pr. Eugene Belilovsky (Co-supervisor of Adel)

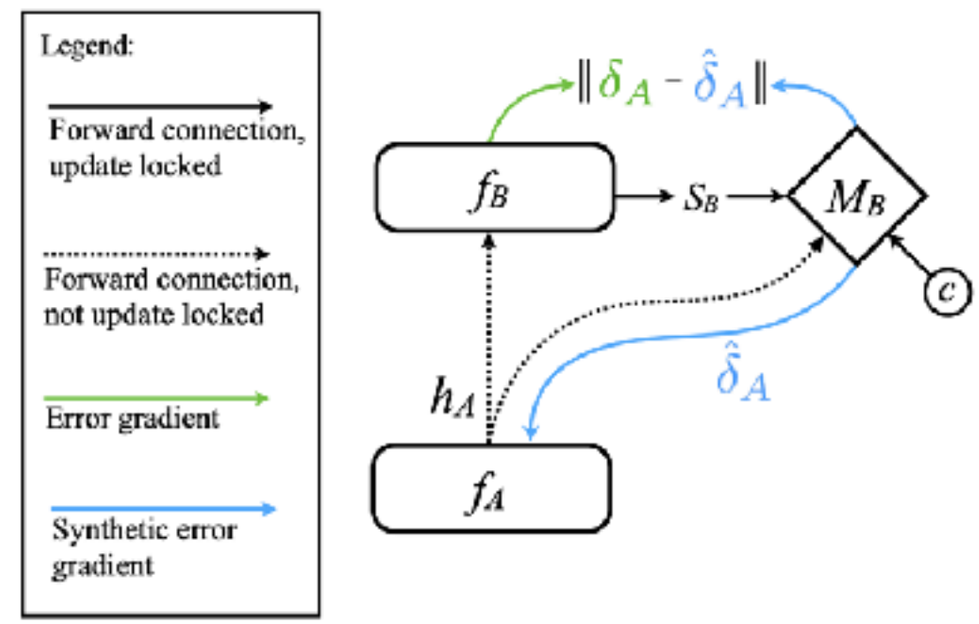
2 publications in:

Alternatives to standard back-prop?

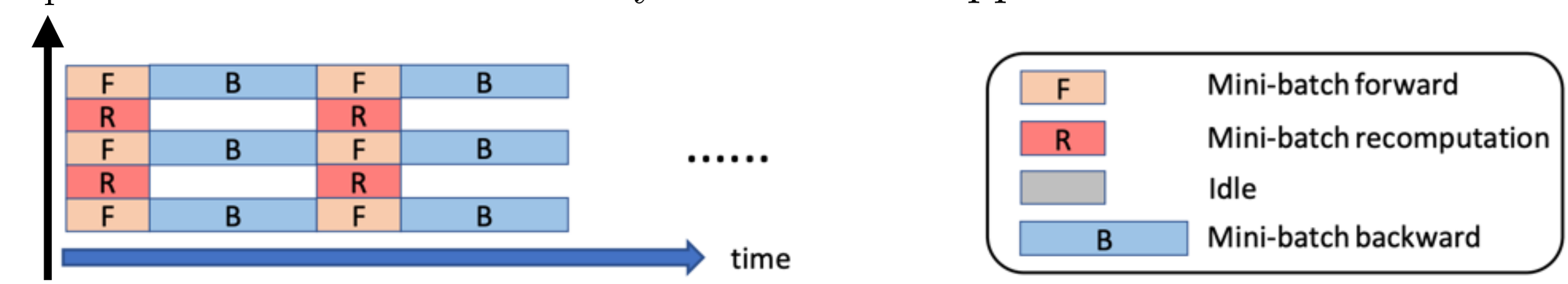
Alternative 1: Direct Feedback Alignment.



Alternative 2: Decoupled Neural Interfaces



Alternative 3: Delayed Gradients approaches

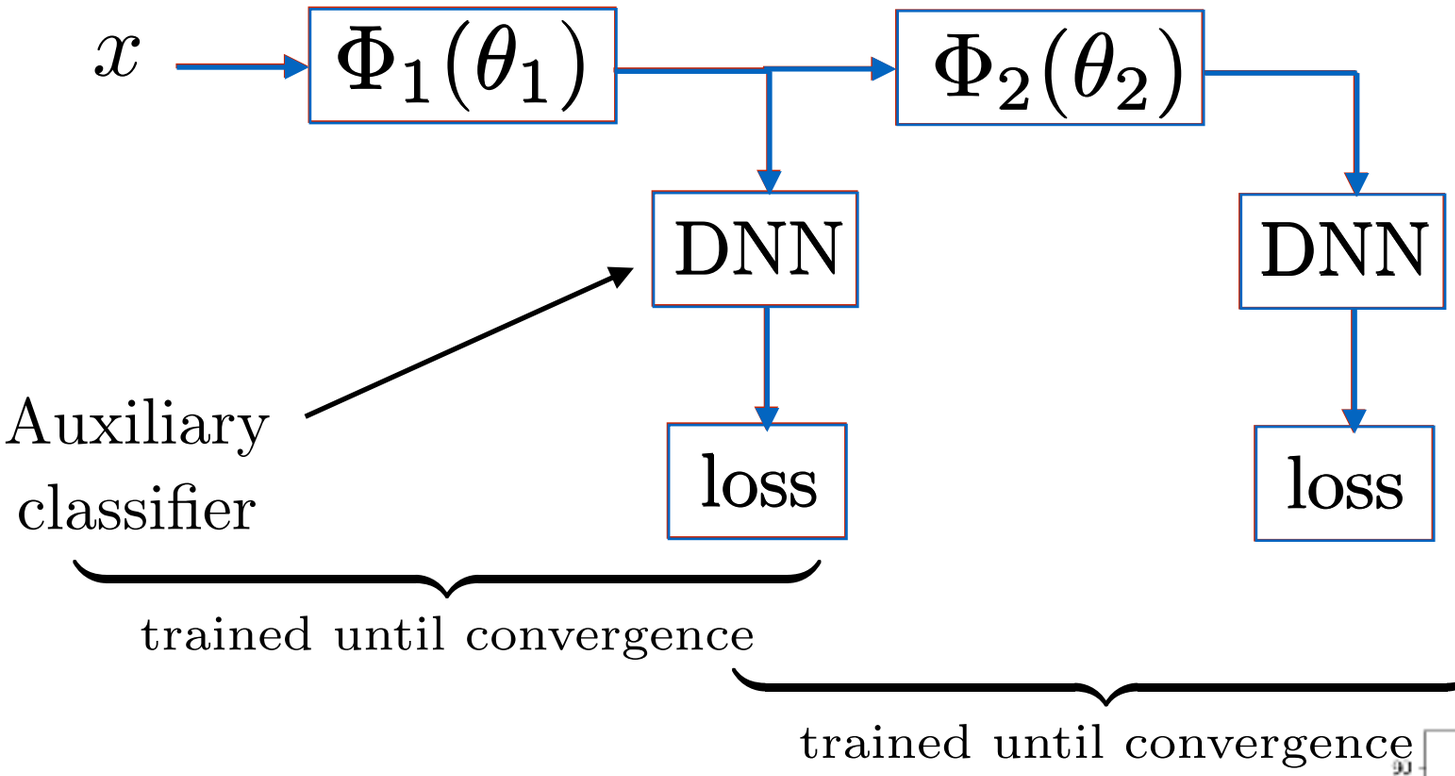


Ref.: **[1]** Direct Feedback Alignment Provides Learning in Deep Neural Networks, Nøkland et al
[2] Decoupled Neural Interfaces using Synthetic Gradients, Jaderberg et al, 2017
[3] On the Acceleration of Deep Learning Model Parallelism with Staleness, Xu et al.

Greedy Learning

Belilovsky E., Eickenberg M. and OE - Greedy Layerwise Learning Can Scale to ImageNet, ICML 2019.

Simply train the CNN layer per layer via supervision...

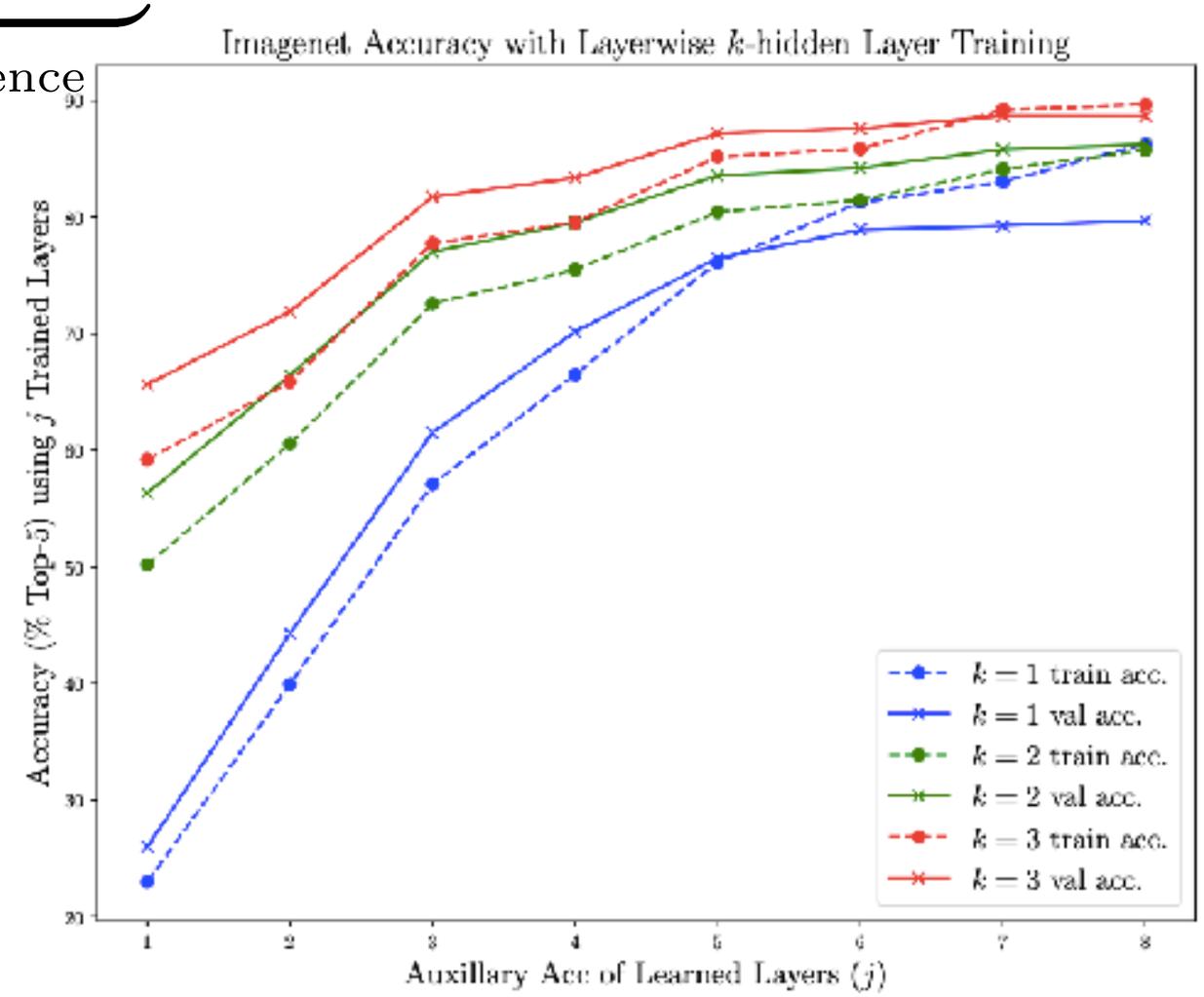


Frozen layers

Trainable layers

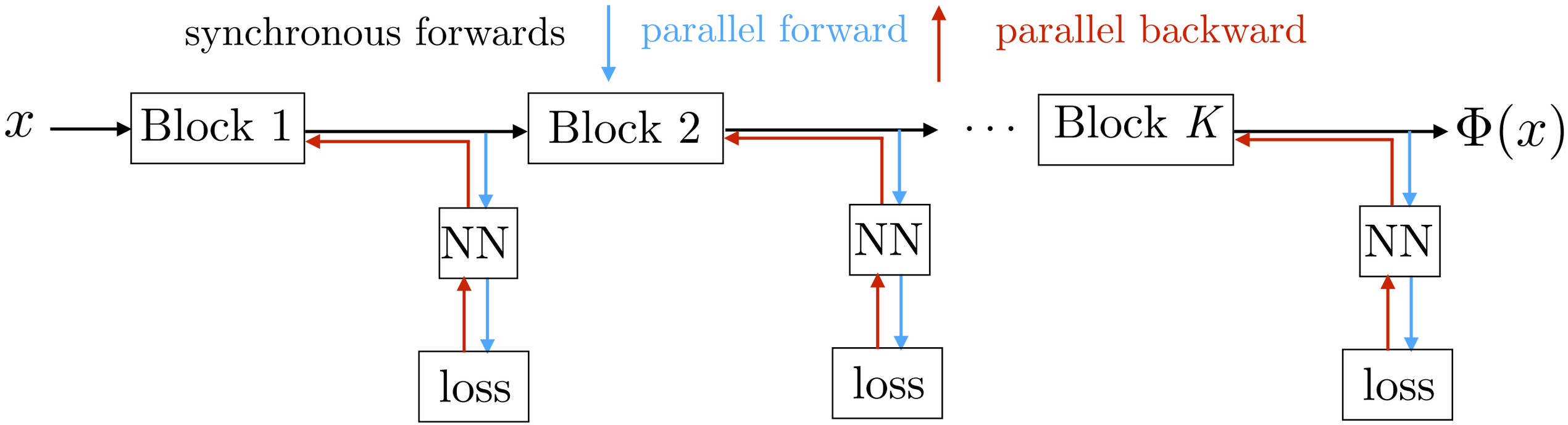
$$\boxed{\text{Let } k = \text{depth(DNN)} + 1}$$

- A very simple idea in the literature for a while...
- But it was not known to not scale!



Decoupled Greedy Learning

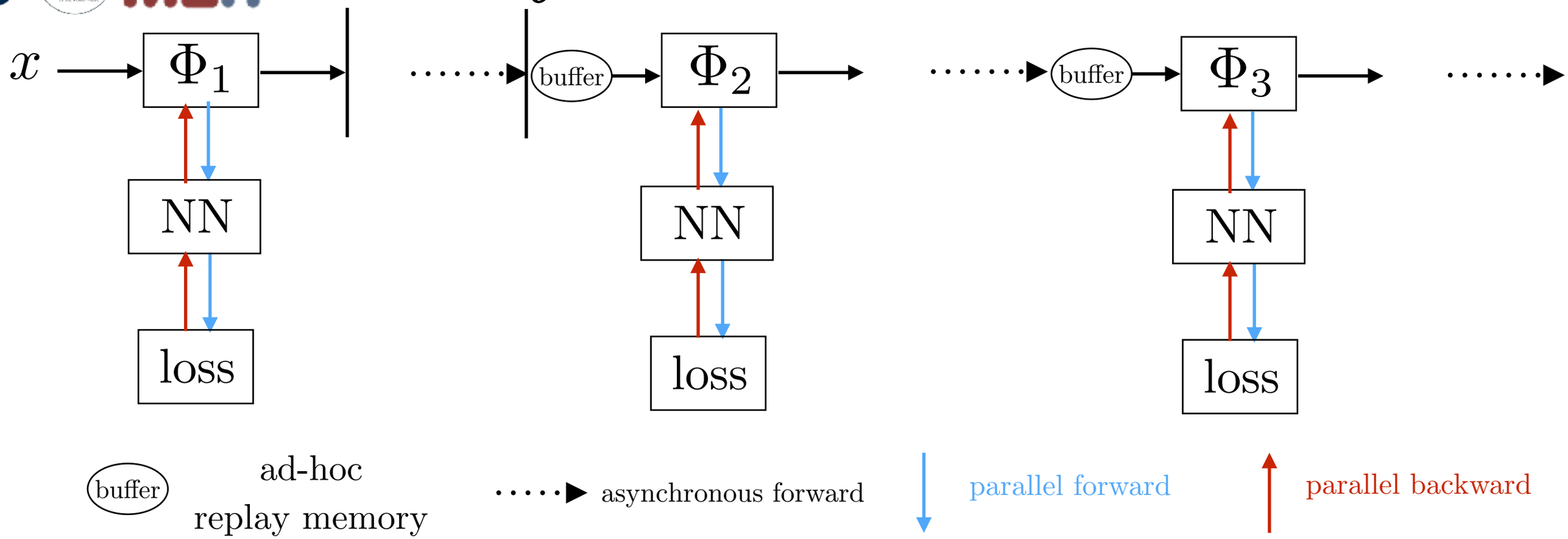
Belilovsky E., Eickenberg M. and OE - Decoupled Greedy Learning of CNNs, ICML 2020.



- Principle: split in K slices a model, and train it.
- Each NN is again small compared to a block.
- It scales on ImageNet!

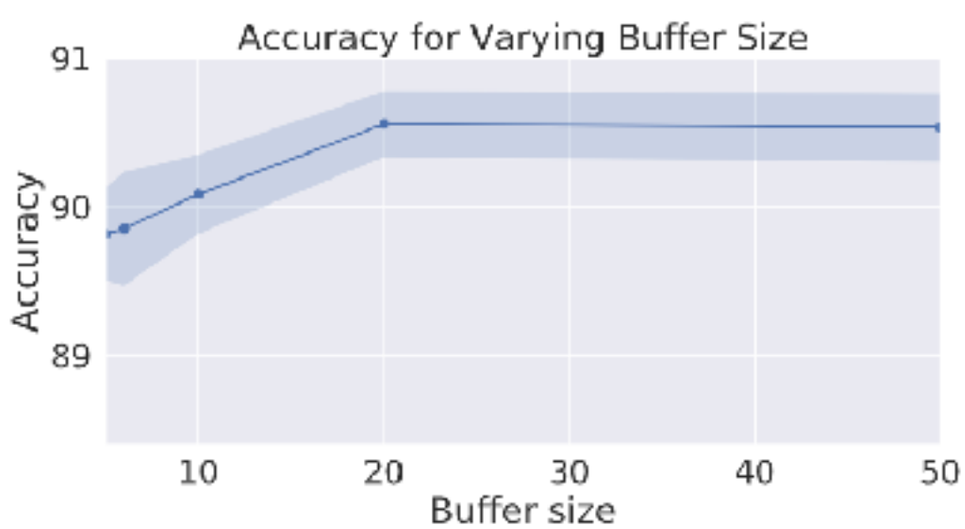
Arch/Perf. on ImageNet	Top 5
VGG-13 (*K*=4)	**88.0**
VGG-13 (*K=13*)	85.8
VGG-13 (end-to-end)	**87.5**
VGG-19 (*K*=4)	89.0
VGG-19 (*K*=2)	**90.2**
VGG-19 (end-to-end)	**89.7**
ResNet-152 (*K*=2)	**92.0**
ResNet-152 (end-to-end)	**92.1**

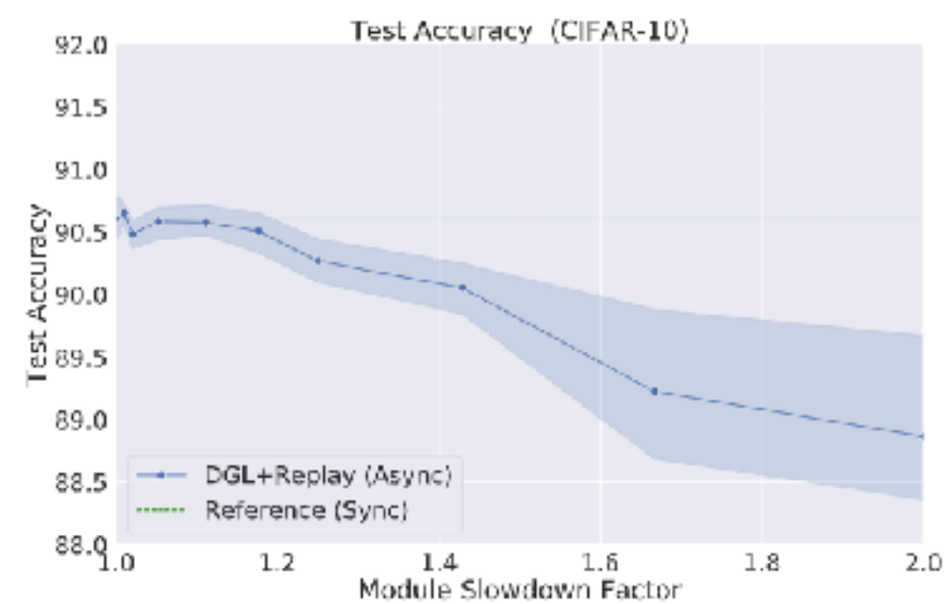
Asynchronous DGL



- We unlock: <u>forward</u>, <u>backward</u> and <u>update</u>.

- Buffers are robust to lags and does not need to be big.





Correcting DGL via Forward Gradient?

Fournier L., Rivaud, S., Belilovsky E., Eickenberg M. and OE - Can Forward Gradient Match Backpropagation?, ICML 2023.

- At a given layer, replace gradient by g obtained from a gradient guess G:

$$x_{t+1} = x_t - \eta_t \nabla f(x_t) \text{ replaced with } x_{t+1} = x_t - \eta_t g(x_t)$$

$$\text{where } g(x) = \langle \nabla f(x), G(x) \rangle G(x) \approx \nabla f(x)$$

Forward gradient

Ref.: Scaling Forward Gradient With Local Losses, Ren et al.

- The stationary points can lead to undesirable solutions...

$$G(x^*) = 0 \quad \text{or} \quad \langle \nabla f(x^*), G(x^*) \rangle = 0$$

- We tested a collection of guesses

Random unbiased estimate:

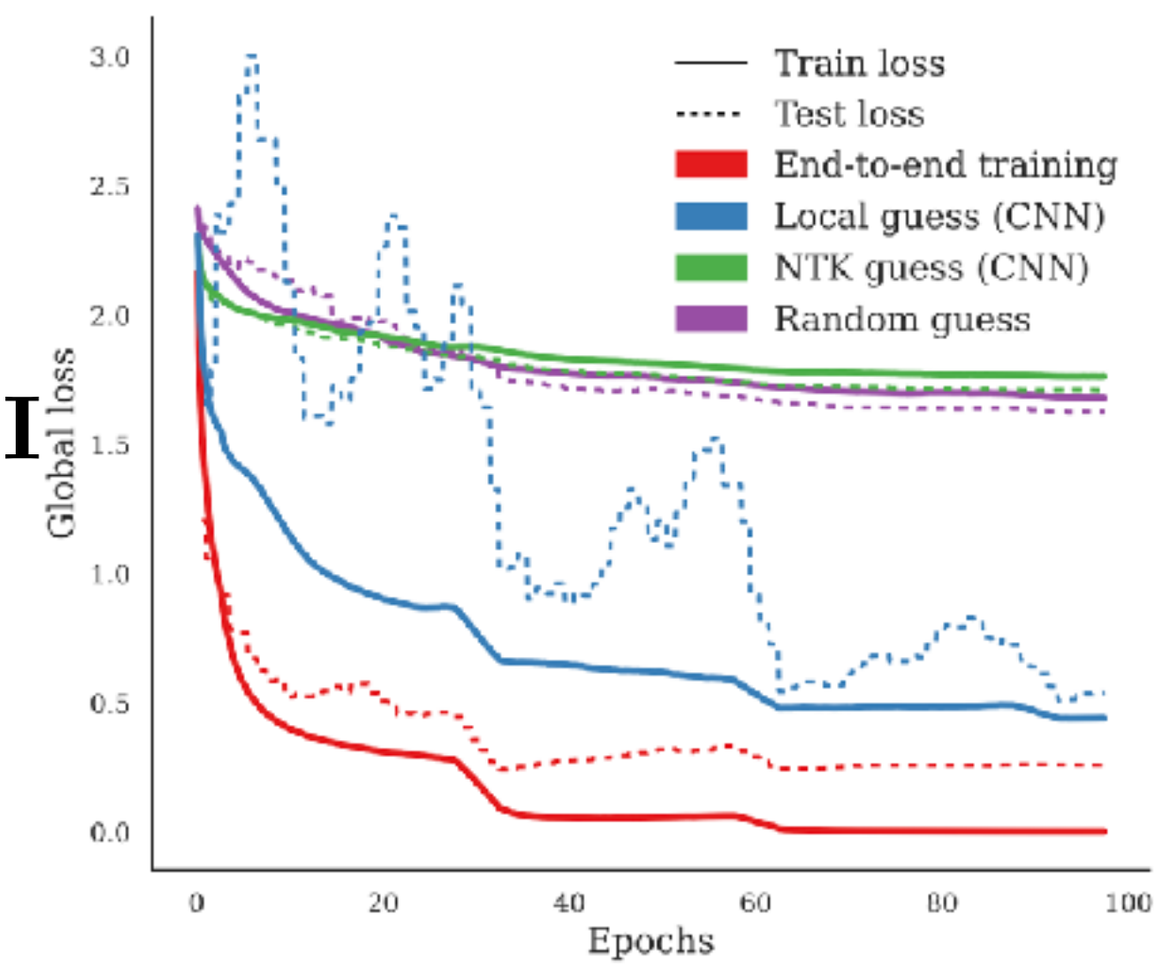
$$\mathbb{E}[G(x)] = 0 \text{ and } \mathbb{E}[G(x)G^T(x)] = \mathbf{I}$$

Randomly initialized DNN:

$$G(x) = \nabla f(x_0), x_0 \sim \mathcal{N}(0, \mathbf{I})$$

Local Auxiliary (DGL):

$$G(x) = \text{AuxDNN}(x, y)$$



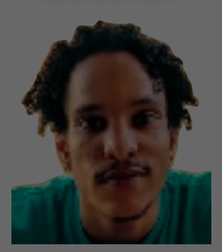
Decoupled Greedy Learning of Deep Neural Networks

(2019-2023)

funded (2021-2022) via ADONIS,

in collaboration with:

Louis Fournier (PhD student)



Dr. Stéphane Rivaud (Postdoc)

Pr. Eugene Belilovsky (Collaborator)

Dr. Michael Eickenberg (Collaborator)

3 publications in:

Asynchronous Decentralized algorithms

(2019-2023)

funded (2021-2022) via ADONIS,

in collaboration with:

Adel Nabli (PhD student)

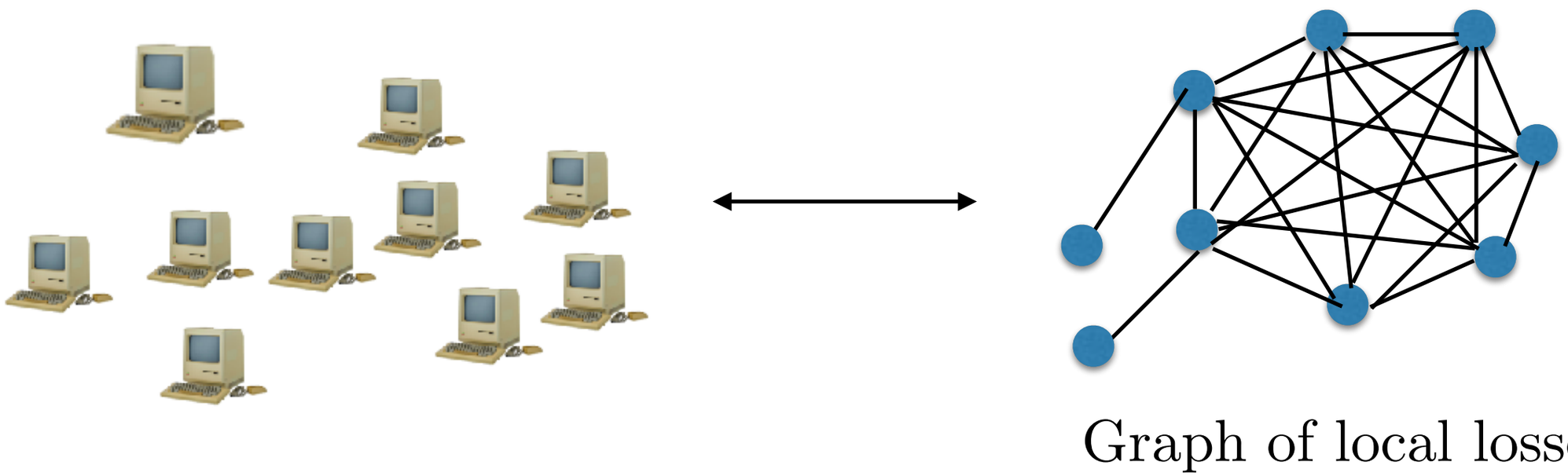
Pr. Eugene Belilovsky (Co-supervisor of Adel)

2 publications in:

Model for Decentralized Learning

Objective: $\inf_{x \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(x)$ with e.g., $f_i(x) = \ell(b_i, x^{\mathrm{T}} a_i) + \frac{\mu}{2}\|x\|^2$

with ℓ convex in x.



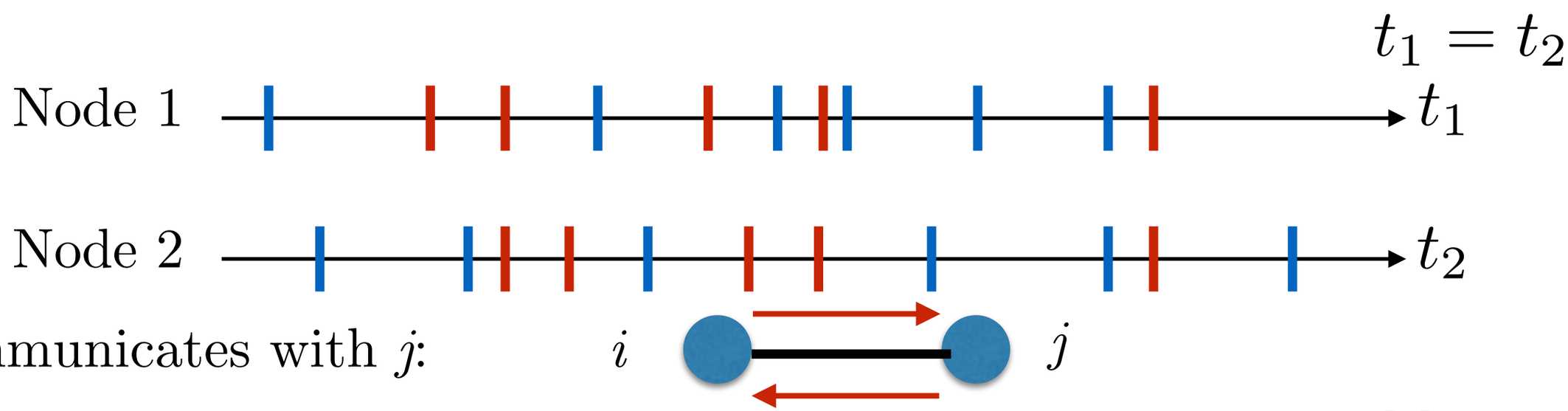
- **Assumption 1**: Local data, ∇f_i and local memory buffer available only on a given node.

- **Assumption 2**: Only adjacent nodes can communicate.

Asynchronous model

- Operations (computations, communications. . .) should happen neither simultaneously, nor in a specific order. (contrary to global **gossips steps**; **wait** barriers, . . .)

- How can we annotate iterates happening at different place, if there is no order?



Assumption 3: The event that i,j communicate is a Poisson process M_{ij} with intensity λ_{ij}.

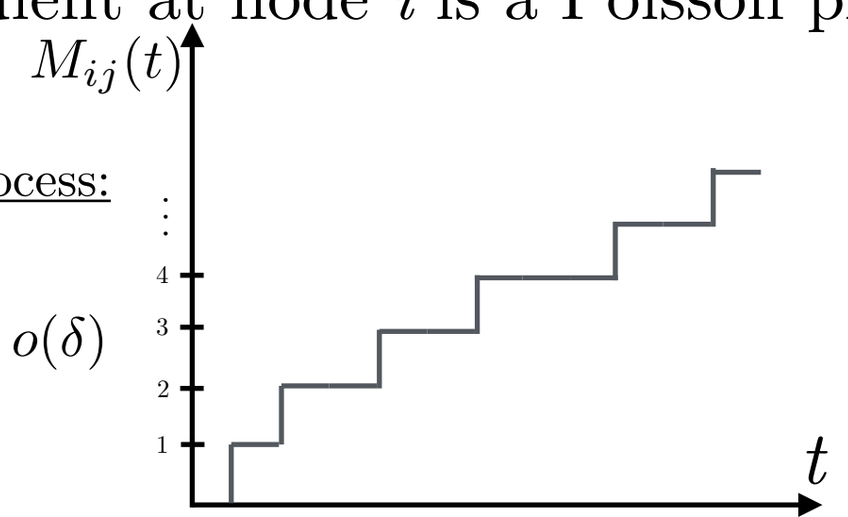
Assumption 4: The event to compute a gradient at node i is a Poisson process N_i with intensity 1.

Ref.: A Continuized View on Nesterov Acceleration for Stochastic Gradient Descent and Randomized Gossip, Even et al., **Best paper Neurips 2022!**

<u>Quick reminder on Pointwise Poisson Process:</u>

For $\delta \to 0^+$ we have

$$\mathbb{P}(M_{ij}(t+\delta) - M_{ij}(t) = 1) = \delta\lambda_{ij} + o(\delta)$$



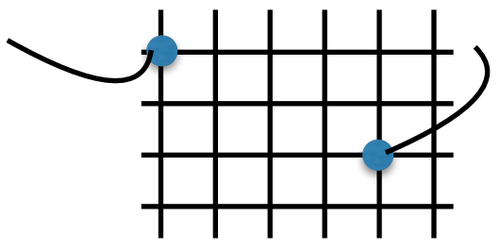
The graph resistance in optimization

Define a (weighted) graph Laplacian via: $\Lambda = \sum_{(i,j)\in\mathcal{E}} \lambda_{ij}(e_i - e_j)(e_i - e_j)^T$

A great quantity to model latency

Ref.: **[1]** Resistance distance. Klein and Randic.
[2] Convex Optimization of Graph Laplacian Eigenvalues, Boyd.
[3] Decentralized Asynchronous Optimization with DADAO allows Decoupling and Acceleration, Nabli and O.

Introduce edge-resistance via: $\chi^{ij} \triangleq \frac{1}{2}(e_i - e_j)^T \Lambda^+ (e_i - e_j)$



literally "electrical" resistance with $R_{ij} = \frac{1}{\lambda_{ij}}$

<u>Maximal resistance:</u> <u>Inverse of the smallest eigen-value:</u>

$$\chi_2 = \frac{1}{2} \sup_{(i,j)\in\mathcal{E}} (e_i - e_j)^T \Lambda^+ (e_i - e_j) \quad \leq \quad \chi_1 = \sup_{\|u\|\leq 1} u^T \Lambda^+ u$$

We will show our rates will depend on $\chi_1, \sqrt{\chi_1 \chi_2}$: can we optimise them?

Proposition (informal, from [2]): For some constraints on i-th node's bandwidth λ_i

$$\begin{aligned} \text{minimize} \quad & \chi_1(\Lambda) \\ \text{subject to} \quad & \sum_i \lambda_{ij} \leq \lambda_i, \lambda_{ij} \geq 0 \ \text{(for } (i,j) \text{ nodes)} \end{aligned}$$

is equivalent to a SDP

Proposition (informal, from [3]): For some constraints on i-th node's bandwidth λ_i

$$\begin{aligned} \text{minimize} \quad & \sqrt{\chi_1(\Lambda)\chi_2(\Lambda)} \\ \text{subject to} \quad & \sum_i \lambda_{ij} \leq \lambda_i, \lambda_{ij} \geq 0 \ \text{(for } (i,j) \text{ nodes)} \end{aligned}$$

is equivalent to a collection of SDP

A simple example: Gossips algorithms

How fast can we compute: $\frac{1}{n}\sum_{i=1}^{n} x_i$ **?**

Ref.: A Continuized View on Nesterov Acceleration for Stochastic Gradient Descent and Randomized Gossip, Even et al., **Best paper Neurips 2022!**

Introduce the spectral gap: $\gamma = \|\Lambda\|\|\Lambda^+\| = \chi_1\|\Lambda\|$

	Synchronous	Asynchronous
Standard	$x^{t+1} = (\mathbf{I} - \alpha\Lambda)^{\gamma} x^t$	$dx_t = -\frac{1}{2}\sum_{(i,j)\in\mathcal{E}} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top x_t dM_{ij}(t)$
Accelerated	$x^{t+1} = P_n(x^t - \alpha\Lambda)x^t$	$\begin{cases} dx_t = \alpha(\tilde{x}_t - x_t)dt - \beta\sum_{(i,j)\in\mathcal{E}} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top x_t dM_{ij}(t) \\ d\tilde{x}_t = \alpha(x_t - \tilde{x}_t)dt - \tilde{\beta}\sum_{(i,j)\in\mathcal{E}} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top x_t dM_{ij}(t) \end{cases}$

Chebychev polynomial

continuous momentum

$$\left(\frac{x_i + x_j}{2}, \frac{x_i + x_j}{2}\right) \to (x_i, x_j)$$

	Synchronous		Asynchronous
Standard	$\mathcal{O}(\gamma)$	$\leq$	$\mathcal{O}(\chi_1)$
		$\not\geq$	
Accelerated	$\mathcal{O}(\sqrt{\gamma})$	$\leq$	$\mathcal{O}(\sqrt{\chi_1\chi_2})$

Asynchronous algorithms are provably better than synchronous algorithms (and more geometric!)

Combining best of both worlds?

Ref.: **[1]** Lower Bounds and Optimal Algorithms for Smooth and Strongly Convex Decentralized Optimization Over Time-Varying Networks, Kovalev et al, 2021
[2] A Continuized View on Nesterov Acceleration for Stochastic Gradient Descent and Randomized Gossip, Even et al.

- [1] proposed an algorithm based on:

$$x^* \in \arg\inf_{x_i = x} \sum_{i=1}^{n} f_i(x_i) \quad \text{is equivalent to} \begin{cases} \nabla f(x^*) - \nu x^* - y^* & = 0 \\ \frac{y^* + \pi z^*}{\nu} + x^* & = 0 \\ \pi z^* + \pi y^* & = 0\,. \end{cases}$$

where $f(x) = (f_i(x_i))_i$ for some $\nu > 0$

Yet, their dynamic relies on an <u>Error Feedback</u> mechanism (which implies a synchronisation between workers) and a <u>synchronous gossip</u> step.

- [2] proposed the following dynamic to minimise $\inf_{x_i = x} \sum_{i=1}^{n} f_i(x_i)$:

$$dx_t^i = \eta(\tilde{x}_t^i - x_t^i)dt - \gamma \sum_{j,(ij)\in\mathcal{E}} (\nabla f_i^*(x_i(t) - \nabla f_j^*(x_j(t))dM_{ij}(t)\,,$$

$$d\tilde{x}_t^i = \eta(x_t^i - \tilde{x}_t^i)dt - \tilde{\gamma} \sum_{j,(ij)\in\mathcal{E}} (\nabla f_i^*(x_i(t) - \nabla f_j^*(x_j(t))dM_{ij}(t)\,.$$

Yet, their dynamic relies on <u>duality</u> and <u>one gradient = one communication</u>, which is inefficient.

DADAO Dynamic

Nabli A. and OE - DADAO: Decoupled Accelerated Decentralized Asynchronous Optimization, ICML 2023.

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

with $\mathbf{N}(t) = (N_i(t))_i$

X_t

Y_t

X_t : gradients

Y_t : communications

which can be compressed into:

$$
\begin{cases}
dX_t = a_1(X_t, Y_t)dt + b_1(X_t)d\mathbf{N}(t) \\
dY_t = a_2(X_t, Y_t)dt + \sum_{(i,j)\in\mathcal{E}(t)} b_2^{ij}(Y_t)dM_{ij}(t)\,,
\end{cases}
$$

- **Theorem** (convergence of DADAO): Under the previous assumptions, if $(f_i)_{i\leq n}$ are μ-strongly convex and L-smooth, then there exists some parameters such that one needs:

- Communications: $\sqrt{\chi_1\chi_2}\, n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$ - Gradients: $n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$

Deterministic case:

- Communications: $\sqrt{\gamma}|\mathcal{E}|\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$ - Gradients: $n\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}$

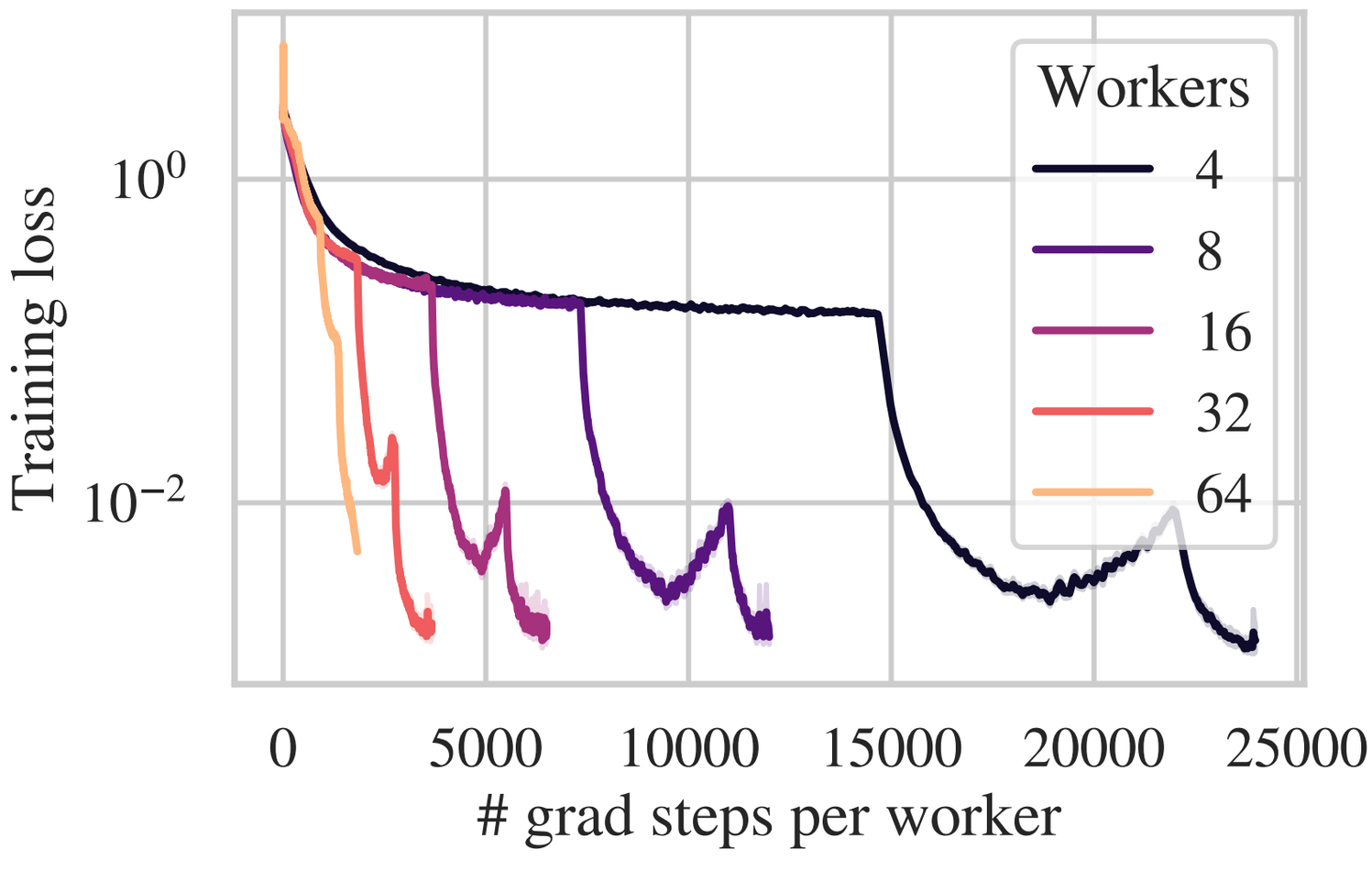
Asynchronous Decentralized Deep Learning

Ref.: Stochastic gradient push for distributed deep learning, Assyrian et al.

- The standard dynamic is given by

$$dx_t^i = -\gamma \int_{\Xi} \nabla F_i(x_t^i, \xi_i)\, dN_t^i(\xi_i) - \frac{1}{2} \sum_{j,(i,j)\in\mathcal{E}} (x_t^i - x_t^j) dM_t^{ij}$$

$$\left(\frac{x^i + x^j}{2}, \frac{x^i + x^j}{2}\right) \to (x^i, x^j)$$

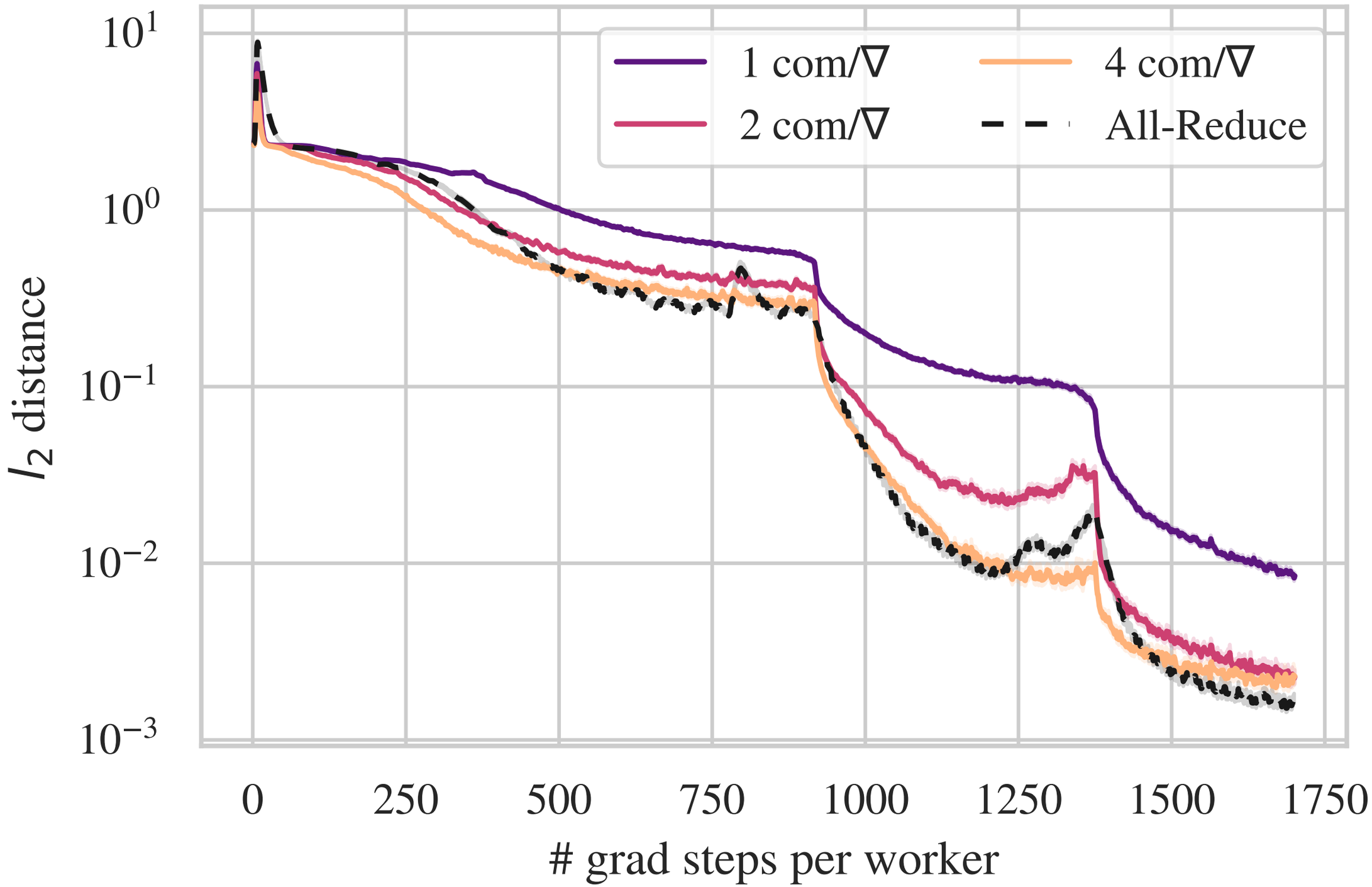


1. Each worker has a copy of CIFAR10 and has access to the whole dataset

2. Total computations & communications are constant accros runs

Ablation on 64 workers

- Why does the method fail with 64 workers? **Hyper-parameters** or **communications**. . . ?



A^2CiD^2: reducing communication rate

Nabli A., Belilovsky E. and OE - A²CiD²: Accelerating Asynchronous Communication in Decentralized Deep Learning, NeurIPS 2023.

- We consider, for hyper-parameters $\eta, \gamma, \alpha, \tilde{\alpha}$, the dynamic given by

$$\begin{cases} dx_t^i = \eta(\tilde{x}_t^i - x_t^i)dt - \gamma \int_{\Xi} \nabla F_i(x_t^i, \xi_i)\, dN_t^i(\xi_i) - \alpha \sum_{j,(i,j)\in\mathcal{E}} (x_t^i - x_t^j) dM_t^{ij} \\ d\tilde{x}_t^i = \eta(x_t^i - \tilde{x}_t^i)dt - \gamma \int_{\Xi} \nabla F_i(x_t^i, \xi_i)\, dN_t^i(\xi_i) - \tilde{\alpha} \sum_{j,(i,j)\in\mathcal{E}} (x_t^i - x_t^j) dM_t^{ij} \end{cases}$$

- Very similar to Slowmo [1], except that this technique (provably) works in a decentralized context.

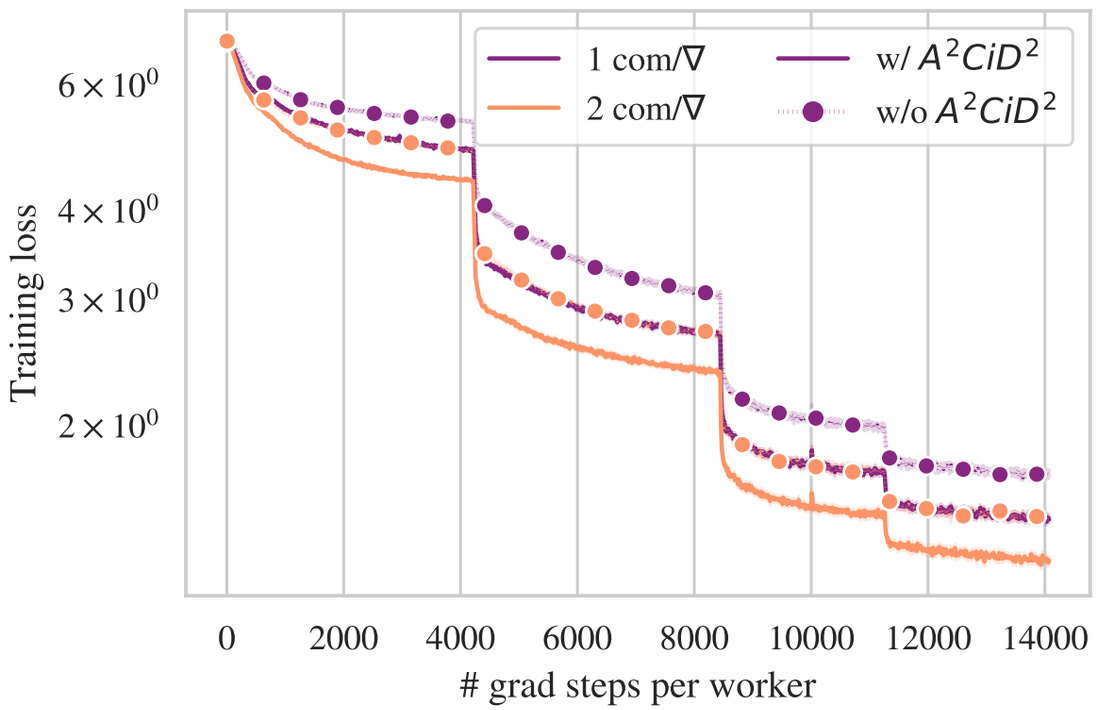
Ref.: **[1]** SlowMo: Improving Communication-Efficient Distributed SGD with Slow Momentum, Wang et al.

- **Note that this dynamic simply needs to double the number of parameters (like momentum SGD) and adds no extra communication.**

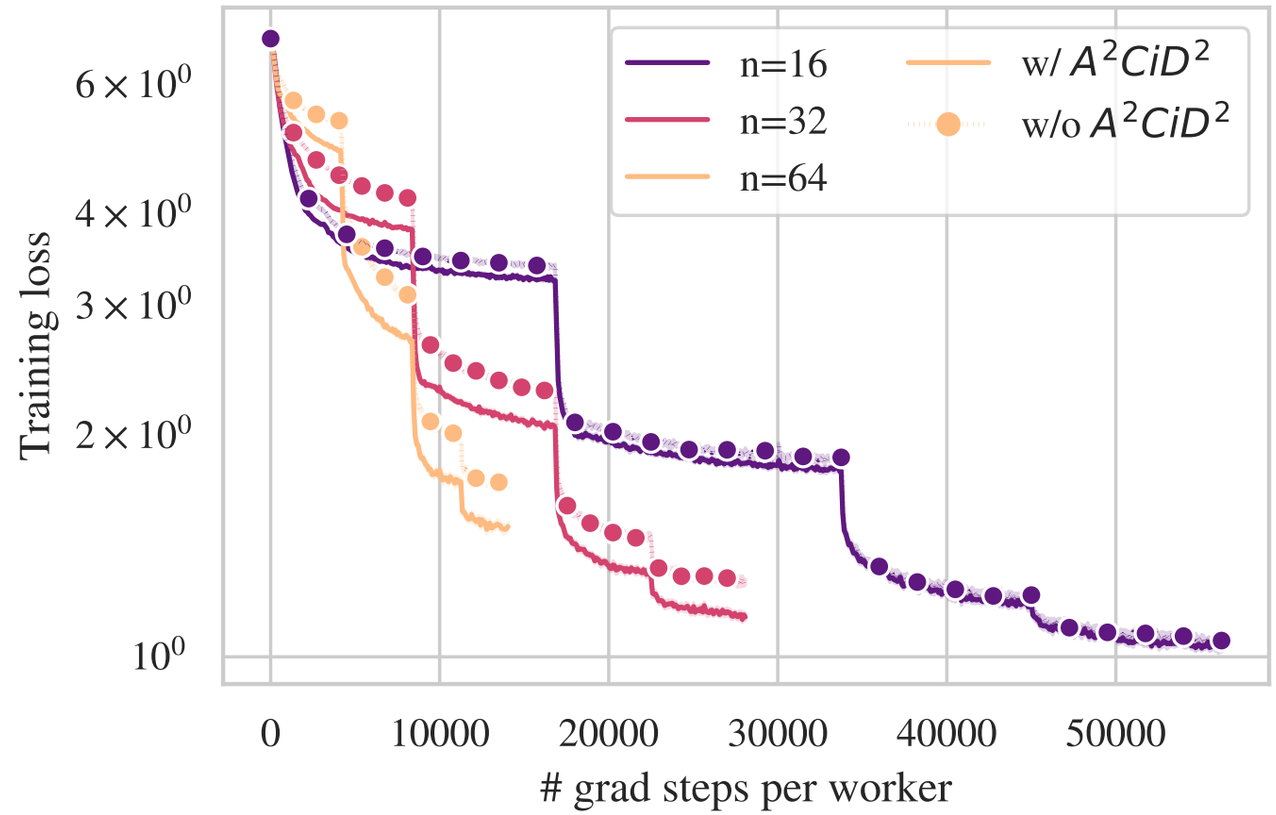
- **Theoretical guarantees**: for SGD in convex/non-convex, replace the connectivity constant χ_1 with $\sqrt{\chi_1\chi_2}$.

Experiments

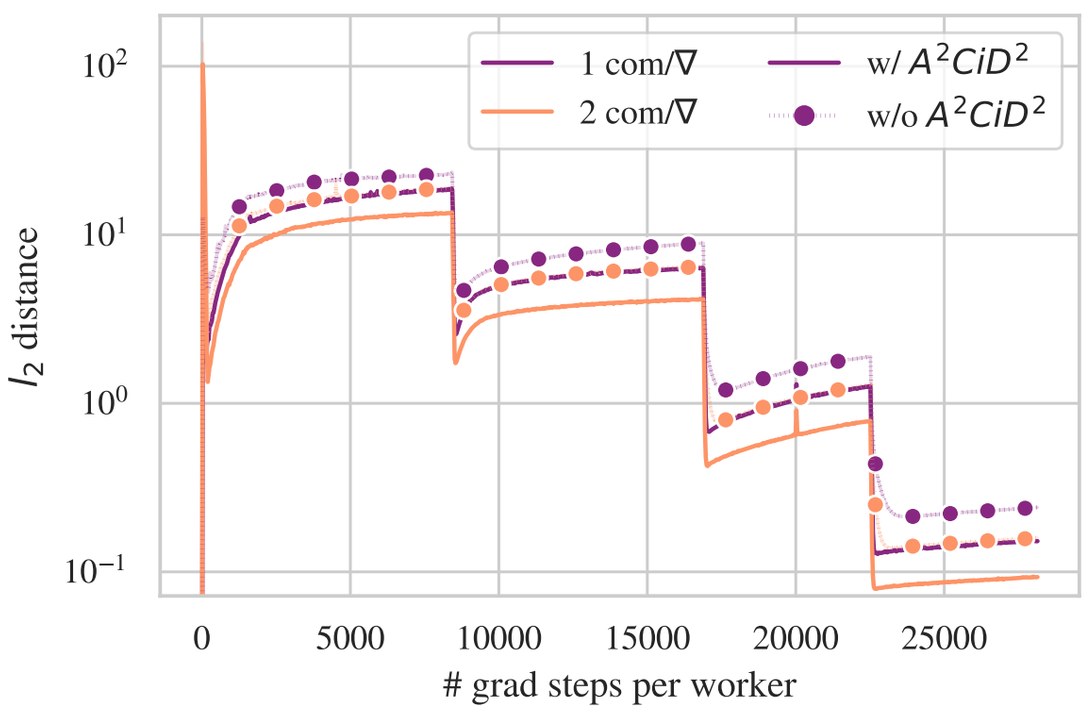
Double VS single communication



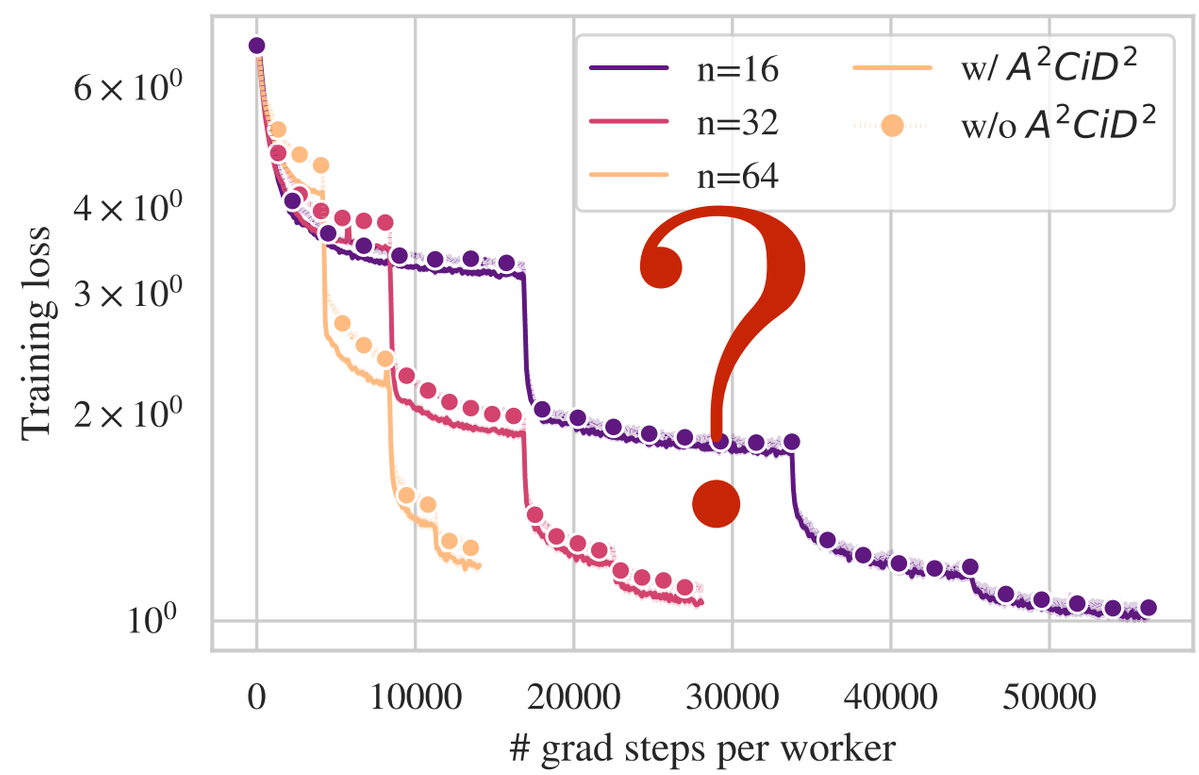
Training loss: ImageNet+ring graph



Consensus distance: double VS single



Training loss: ImageNet+complete graph



What about generalisation?

#Workers	4	8	16	32	64
AR-SGD baseline	94.5±0.1	94.4±0.1	94.5±0.2	93.7±0.3	92.8±0.2
Complete graph					
Async. baseline	94.9±0.1	94.9±0.07	94.9±0.04	94.7±0.02	93.4±0.2
Ring graph					
Async. baseline	**95.0**±0.06	**95.0**±0.01	**95.1**±0.07	94.4±0.04	91.9±0.10
A^{2}CiD^2	94.9±0.02	**95.0**±0.1	95.0±0.2	**95.0**±0.08	**92.9**±0.2

CIFAR10

#Workers	#com/#grad	16	32	64
AR-SGD baseline	-	75.5	75.2	74.5
Complete graph				
Async. baseline	1	74.6	73.8	71.3
A^{2}CiD^2	1	75.1	**74.5**	72.6
A^{2}CiD^2	2	**75.2**	74.4	**74.3**
Ring graph				
Async. baseline	1	**74.8**	71.6	64.1
A^{2}CiD^2	1	74.7	**73.4**	**68.0**
Async. baseline	2	74.8	73.7	68.2
A^{2}CiD^2	2	**75.3**	**74.4**	**71.4**

ImageNet

Adel Nabli
(PhD student)

A very efficient implementation allows to efficiently run jobs and get an actual speed-up.

	n	4	8	16	32	64
Ours	t (min)	**20.9**	**10.5**	**5.2**	**2.7**	**1.5**
AR	t (min)	21.9	11.1	6.6	3.2	1.8

Part 4: Conclusions and perspectives on distributed learning

Conclusions

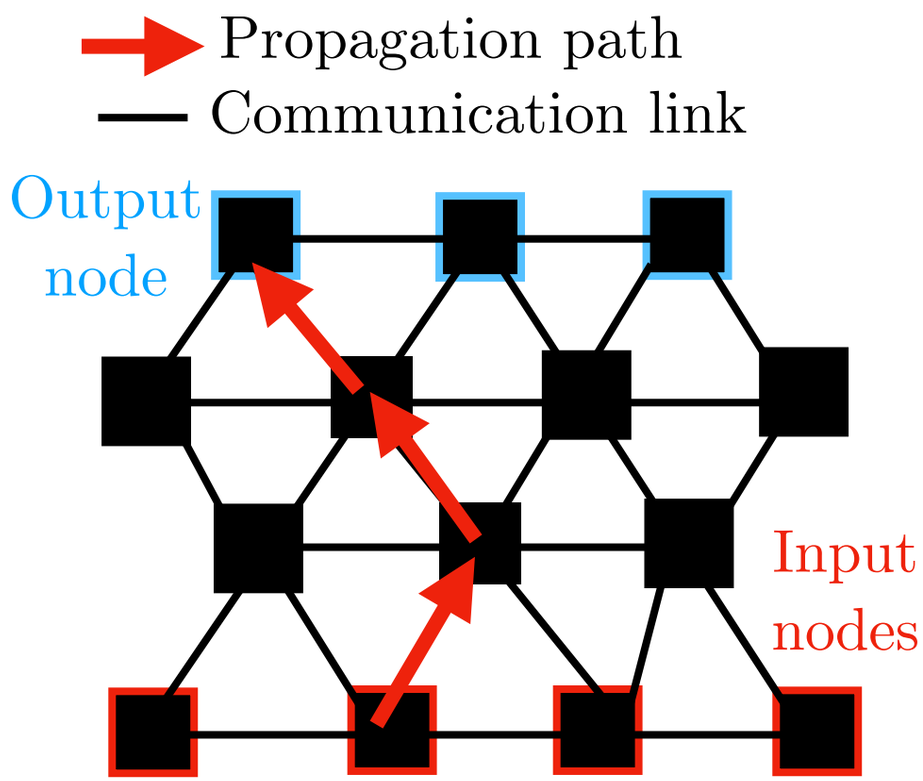
- **Decoupling, randomising** are keys tools to propose novel, faster algorithms which use better hardware to train Deep Neural Networks.

- **<u>However significant limits remain to practically use those algorithms:</u>**

  - Lack of practical implementations
  - Applications are restricted to vision task/linear models.
  - Accuracy gaps are significant.

Perspectives for Local Learning

- Toward swarm of layers: local SGD + local Learning.



- Alternatives to Greedy Learning, which scale better?

- Local losses for generative models?

Perspectives for Asynchronous Decentralized Learning

- Collaborative training with guarantees?
- Mitigating the Need for grid search?
- Filling in this Table of lower bounds for smooth strongly convex functions:

	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.	?	$\lvert\mathcal{E}\rvert \sqrt{\gamma} \log \frac{1}{\epsilon} \sqrt{\frac{L}{\mu}}$	?	?

Future?

Collaborative frameworks for large scale machine learning applied to science

- **Combining** academic **resources** and non academic (*e.g.*, webGPU by apache) to be able to train very large models collaboratively on open data (*grants submitted*).

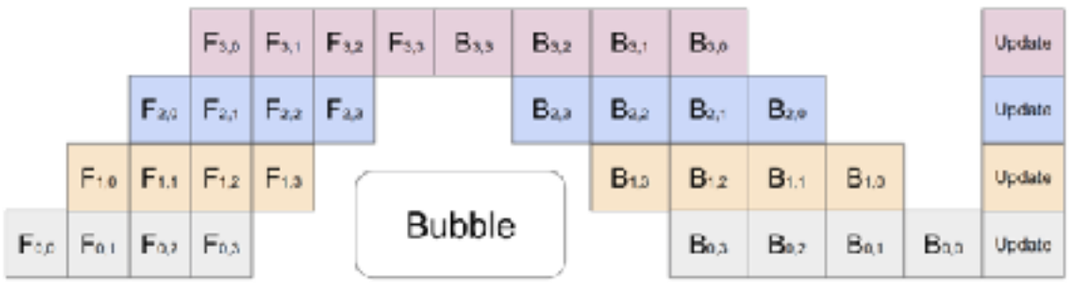
- It could be a cornerstone of a broader approach that could be applied to science in general: instead of other disciplines incorporating AI, **AI is being integrated into these disciplines**. (*sabbatical at Flatiron Institute*)

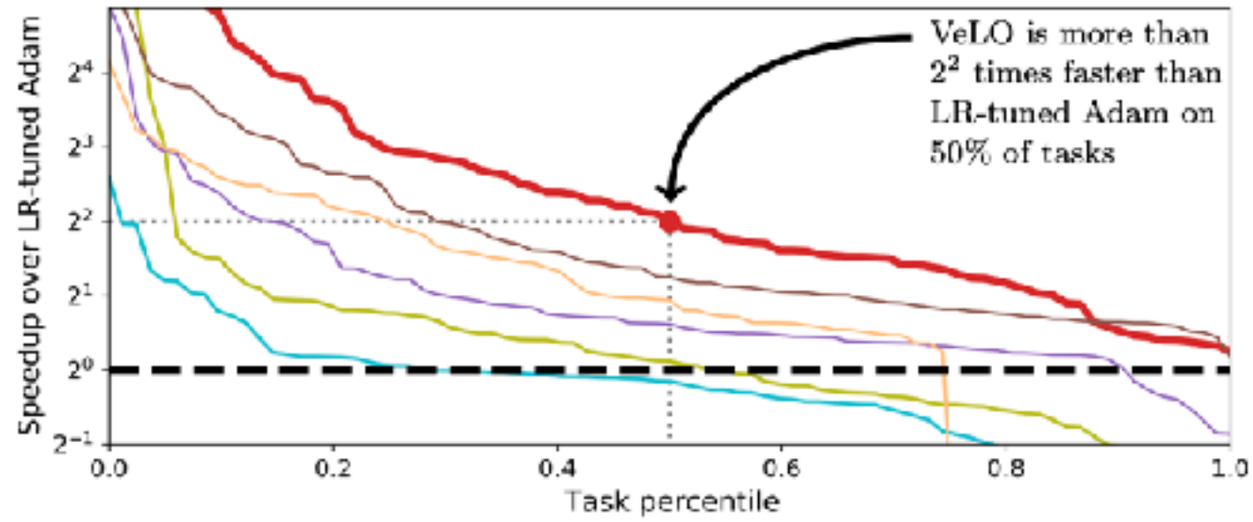
Annex

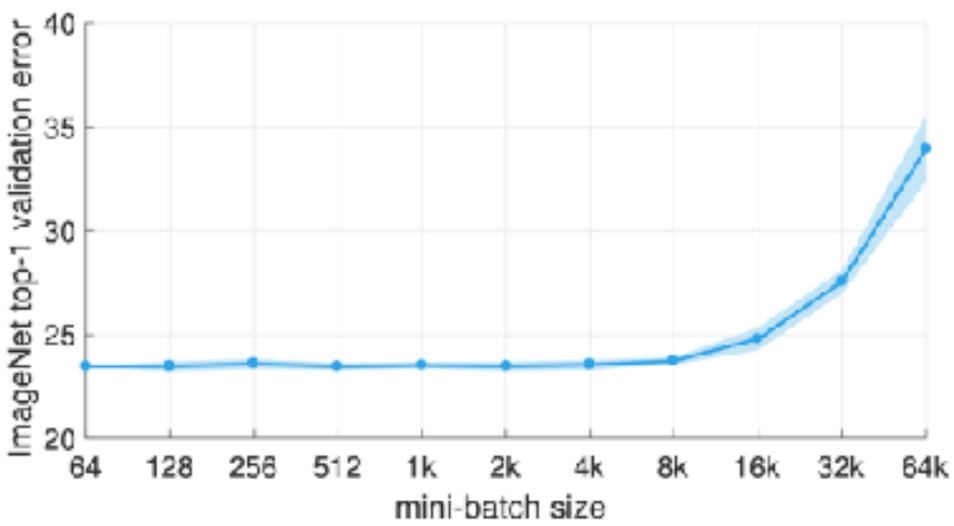
Improving distributed training of Deep Neural Networks

Strategy 1: Pipelining

(handling slices of large models)

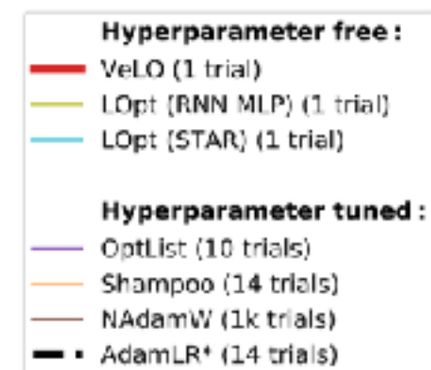






Strategy 2: Self-optimizers

(reducing grid search dependency)



Strategy 3: Zero-DP

(shards of model weights)

Strategy 4: Large batches

(achieving higher data-parallelism)

Strategy 5: Hogwild!

(asynchronous centralized updates)

Algorithm 1 HOGWILD! update for individual processors

```
loop
    Sample e uniformly at random from E
    Read current state x_e and evaluate G_e(x_e)
    for v ∈ e do x_v ← x_v − γ G_ev(x_e)
end loop
```

Ref.: **[1]** GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism, Huang et al.
[2] VeLO: Training Versatile Learned Optimizers by Scaling Up, Metz et al.
[3] ZeRO: Memory Optimizations Toward Training Trillion Parameter Models, Rajbhandari et al.
[4] Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour, Goyal et al.
[5] Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent, Recht et al.

Convergence guarantees

- Assumption (a): each f_i is μ-strongly convex and L smooth:

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

- Assumption (b): each f_i is L smooth and:

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

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

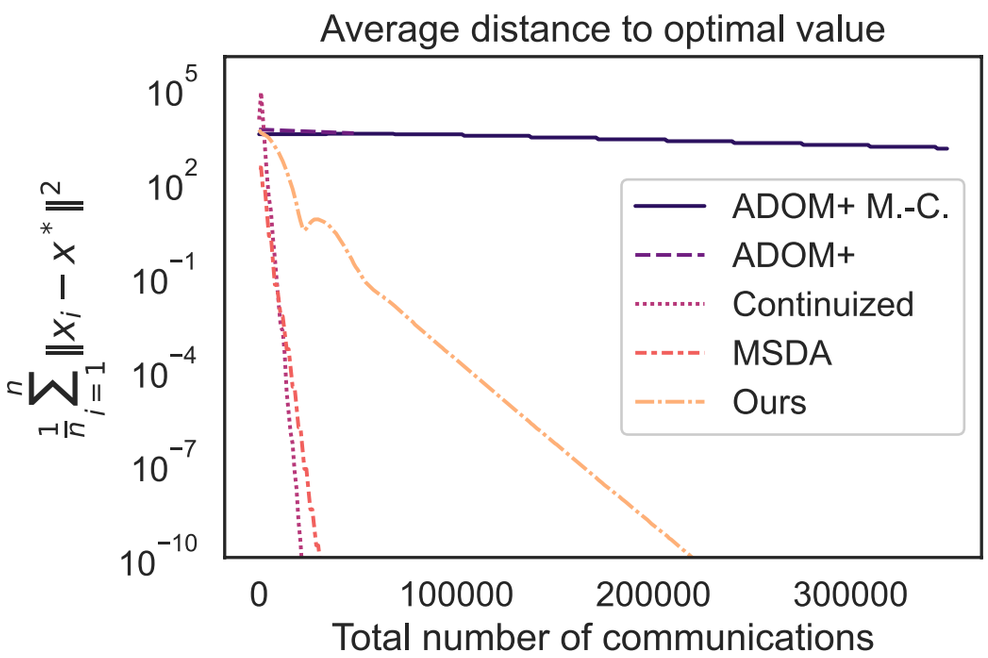
Here: $\chi_2 \triangleq \sup_{(i,j)\in\mathcal{E}}\frac{1}{2}(e_i-e_j)^T\Lambda^+(e_i-e_j)\leq\chi_1=\sup_{\|u\|=1}u^T\Lambda^+u$

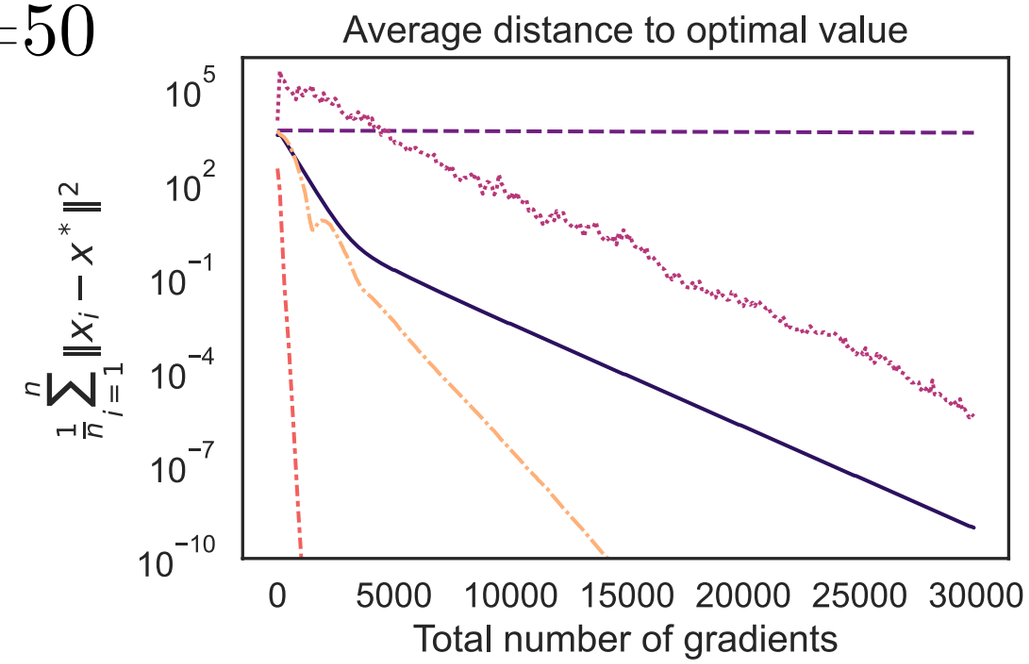
Method	Strongly Convex	Non-Convex
Koloskova et al. [21]	$\frac{\sigma^2}{n\mu^2\epsilon}+\sqrt{L}\frac{\chi_1\xi+\sqrt{\chi_1}\sigma}{\mu^{3/2}\sqrt{\epsilon}}+\frac{L}{\mu}\chi_1$	$\frac{L\sigma^2}{n\epsilon^2}+L\frac{\chi_1\xi+\sqrt{\chi_1}\sigma}{\epsilon^{3/2}}+\frac{L\chi_1}{\epsilon}$
A^{2}CiD^2 (Ours)	$\frac{\sigma^2+\sqrt{\chi_1\chi_2}\xi^2}{\mu^2\epsilon}+\frac{L}{\mu}\sqrt{\chi_1\chi_2}$	$L\frac{\sigma^2+\sqrt{\chi_1\chi_2}\xi^2}{\epsilon^2}+\frac{L\sqrt{\chi_1\chi_2}}{\epsilon}$
AD-PSGD [27]	-	$L\frac{\sigma^2+\xi^2}{\epsilon^2}+\frac{nL\chi_1}{\epsilon}$

Numerical experiments

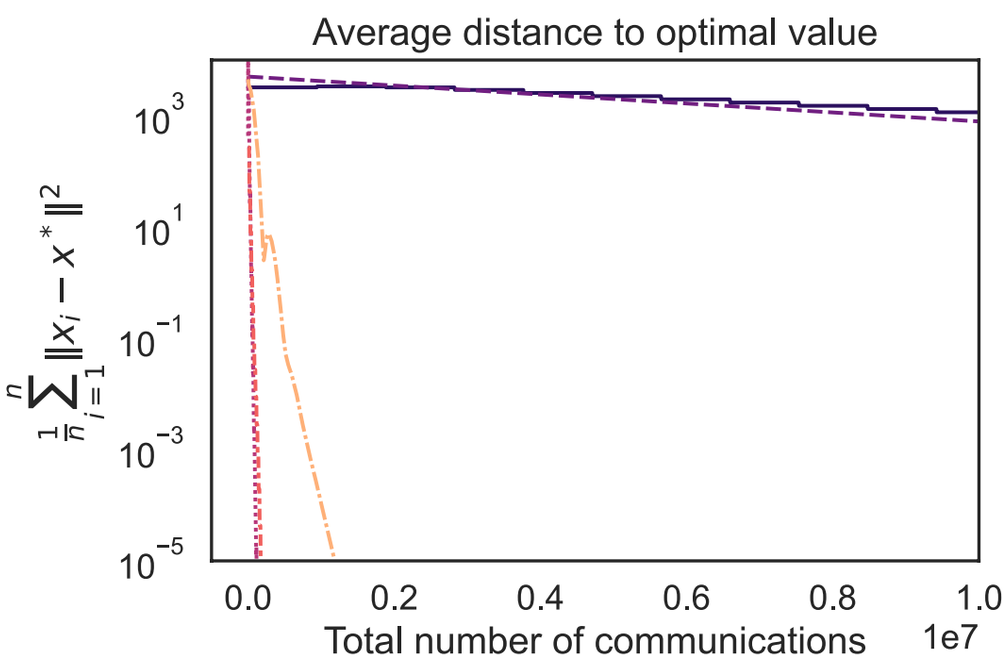
(logistic+linear regression)

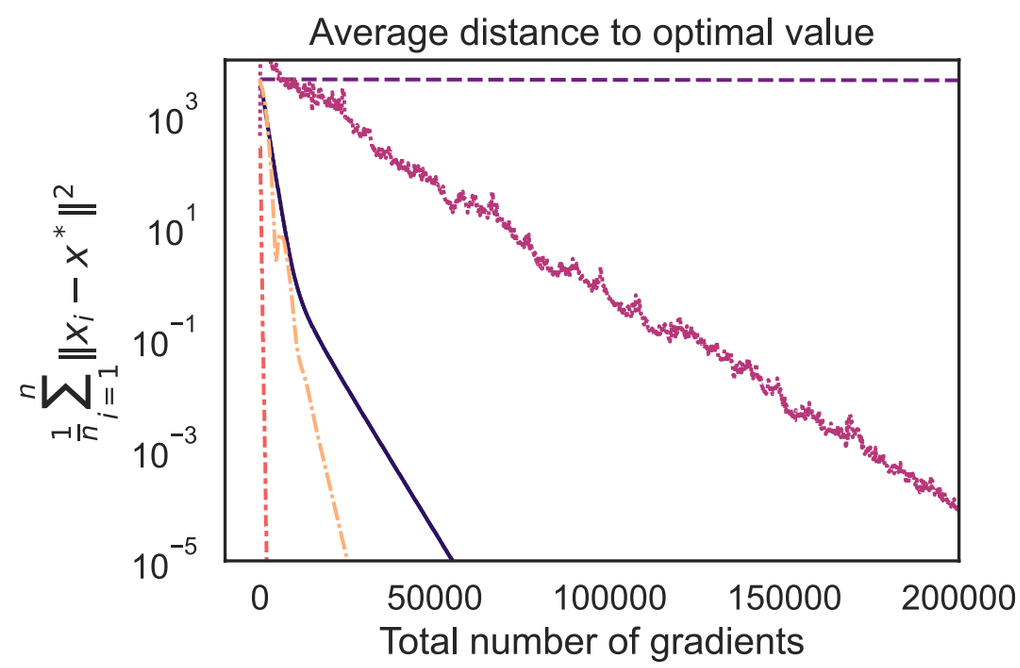
Line n=50





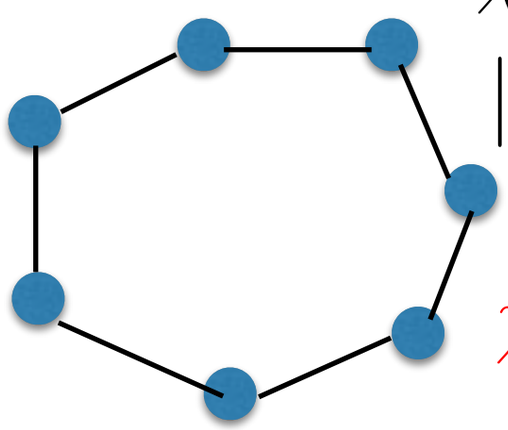
Line n=150





Spectrum of well-known graphs

Line/circle



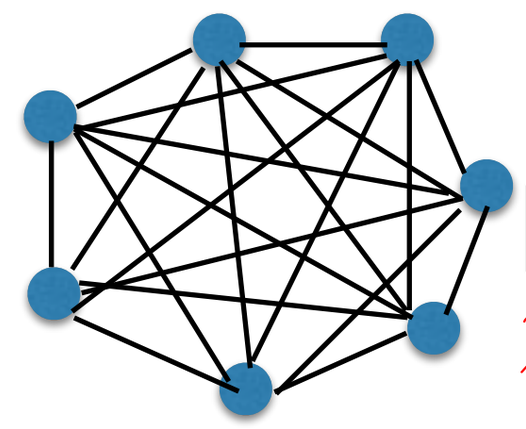
$\chi_1 = n^2$

$\|\Lambda\| = 1$

$|\mathcal{E}| = n$

$\chi_2 = 1$

Complete



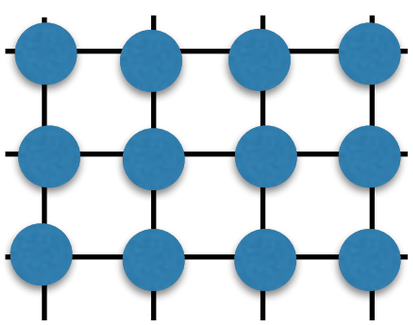
$\chi_1 = 1$

$\|\Lambda\| = 1$

$|\mathcal{E}| = n^2$

$\chi_2 = 1$

d-connectivity



$\chi_1 = n^{2/d}$

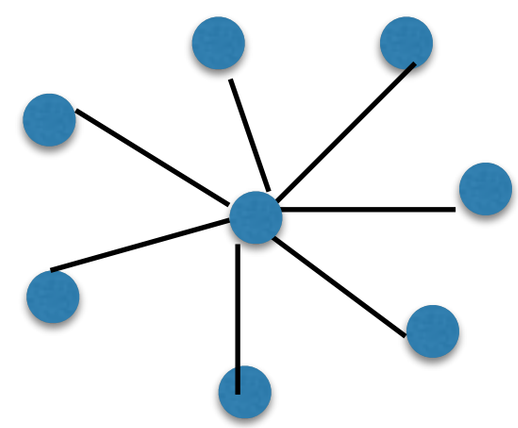
$\|\Lambda\| = 1$

$|\mathcal{E}| = dn$

$\chi_2 = 1$

How can we benefit the effective resistance?

Star-graph



$\chi_1 = 1$

$\|\Lambda\| = n$

$|\mathcal{E}| = n$

$\chi_2 = 1$