Multi-GPU distributed PnP-ULA for high-dimensional imaging inverse problems

Maxime Bouton*, Pierre-Antoine Thouvenin*, Audrey Repetti†‡, and Pierre Chainais*
*Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
†School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh EH14 4AS, UK
‡Maxwell Institute for Mathematical Sciences, Edinburgh EH8 9BT, UK

***Abstract*—Markov Chain Monte Carlo (MCMC) methods enable uncertainty quantification in inverse problems, making them valuable in various applications, especially when no ground-truth is available. Optimization-inspired Plug-and-Play MCMC algorithms, like PnP-ULA, have been developed to incorporate rich neural networks as priors, improving drastically the estimation quality. However, scaling MCMC samplers remains challenging, as generating and storing a sufficient number of high-dimensional samples, typically high-resolution images, can be computationally prohibitive. This work proposes a distributed implementation of PnP-ULA to target much larger problems without compromising on estimation quality. The proposed approach leverages a lightweight deep denoiser within a Single Program Multiple Data that permits an efficient exploitation of a multi-GPU architecture. Synthetic imaging tasks demonstrate the scalability and performance of this strategy to deal with very high dimensional inverse problems. This approach achieves competitive estimation quality along with uncertainty quantification compared to a typical state-of-the-art PnP optimization method, with much higher scalability.**

***Index Terms*—inverse problems, Markov chain Monte Carlo (MCMC), distributed multi-GPU computing, PnP algorithm**

I. Introduction

This work focuses on the resolution of high-dimensional imaging inverse problems, which aim at recovering an unknown latent image $\overline{\boldsymbol{x}} \in \mathbb{R}^N$ from a set of degraded and noisy observations $\boldsymbol{y} \in \mathbb{R}^M$. For such applications, both N and M are usually large, typically between 10^3 and 10^7. The relationship between observations and parameters is defined by an observation model of the form

$$\boldsymbol{y} = \mathcal{A}(\boldsymbol{H}\overline{\boldsymbol{x}}), \tag{1}$$

where $\boldsymbol{H} \in \mathbb{R}^{M\times N}$ represents the measurement operator, and $\mathcal{A} : \mathbb{R}^M \to \mathbb{R}^M$ models random perturbations such as noise affecting the measurements. Combined with a prior, the associated likelihood leads to a posterior distribution whose density is typically of the form

$$\pi(\boldsymbol{x} \mid \boldsymbol{y}) \propto \exp\left(-\phi_{\boldsymbol{y}}(\boldsymbol{H}\boldsymbol{x})\right) p(\boldsymbol{x}), \tag{2}$$

This work was supported by the ANR project "Chaire IA Sherlock" ANR-20-CHIA-0031-01 hold by P. Chainais, the national support within the *programme d'investissements d'avenir* ANR-16-IDEX-0004 ULNE, Région HDF, and the CNRS IEA "DAISHI" hold by P.-A. Thouvenin and A. Repetti. The project was provided with computing HPC and storage resources by GENCI at IDRIS on the supercomputer Jean Zay's V100 partition thanks to the grant 2024-AD010615597.

where the data-fidelity term $\phi_{\boldsymbol{y}} : \mathbb{R}^M \to]-\infty, +\infty]$ is additively separable, $\phi_{\boldsymbol{y}} \circ \boldsymbol{H}$ is L_1-Lipschitz differentiable, and $p : \mathbb{R}^N \to [0, 1]$ is the density of the prior distribution.

The choice of the prior plays a crucial role to form high-quality estimates. Traditional handcrafted priors, e.g., based on the total variation or alternatives [1]–[4] aimed at promoting sparsity in a transformed domain, have shown limitations to capture complex image structures. Training a denoiser encoded by a neural networks has emerged as a powerful way to convey prior information within standard inference algorithms. This approach is referred to as Plug-and-play (PnP) [5]–[7]. Among the denoisers investigated in the literature, unfolded algorithms [8] such as the Deep Dual Forward-Backward (DDFB) [9]–[11] follow a principled approach to design network architectures. Inspired by the structure of iterative optimization algorithms, unrolled networks can achieve competitive denoising performance with fewer parameters compared to state-of-the-art alternatives such as DRUNet [12]. In practice, lightweight denoisers are key towards scalable PnP algorithms, compromising between reconstruction quality, inference speed and scalability. PnP methods have been widely investigated in the optimization literature [13], and later studied within Markov chain Monte Carlo (MCMC) algorithms [14].

Uncertainty quantification is critical to interpret estimates, especially in absence of ground truth data. Unlike optimization methods, MCMC algorithms provide not only point estimates, but also permit to quantify associated uncertainties. Following the same principles as PnP optimization algorithms, PnP samplers have been recently proposed in the literature, such as the Plug-and-Play Unadjusted Langevin Algorithm (PnP-ULA) [14] and the Plug-and-Play Split Gibbs Sampler (PnP-SGS) [15]. Under a few technical assumptions, there exists a proper prior distribution associated with the denoiser [14].

Distributed computing, i.e., leveraging multiple computing units (CPUs or GPUs) referred to as workers, is a usual approach to scale algorithms to high-dimensional problems. In particular, MCMC algorithms are sequential in nature, which can limit their deployment on distributed architectures. A solution to implement distributed MCMC algorithms is to decompose each computing task over multiple workers, following the Single Program Multiple Data (SPMD) paradigm [16]. The price to pay is the necessary communications between some of the workers before part of the overall computing task that are locally carried out by each worker. Many operations

such as convolutions, typically in image processing, only require a small number of communications to be implemented within an SPMD architecture. As a consequence, a distributed implementation is expected to be computationally efficient.

Taking inspiration from [17], this paper introduces a fully distributed MCMC algorithm leveraging a lightweight DDFB denoiser under the usual assumption that $\phi_{\boldsymbol{y}}$ in (2) is additively separable. The algorithm is based on the PnP-ULA framework, but can be seamlessly adapted to other Langevin-based PnP transition kernels. To effectively handle very large inverse problems, the proposed implementation exploits multiple GPUs, ensuring high computational performance and scalability. The proposed distributed algorithm is equivalent to its sequential counterpart up to the behaviour of the pseudo-random number generator. Thus it benefits from the same convergence guarantees as PnP-ULA.

This paper is organized as follows. Section II outlines the limitations of a serial implementation of PnP-ULA for handling very large problems and introduces the proposed distributed algorithm. Section III illustrates the restoration performance and scalability of this algorithm on large deconvolution problems in the presence of additive white Gaussian noise. Section IV summarizes the related conclusions.

II. Model structure and proposed approach

This section outlines how PnP-ULA [14] can be efficiently distributed. Under a few assumptions, the methodology detailed in [17] to distribute simpler priors can be adopted for specific deep denoisers.

A. Towards a distributed PnP-ULA implementation

To draw samples from (2), the PnP-ULA transition from state $t \in \mathbb{N}$ to $t+1$ is defined by [14]

$$\begin{aligned}\boldsymbol{x}^{(t+1)} &= \boldsymbol{x}^{(t)} - \delta \boldsymbol{H}^* \nabla \phi_{\boldsymbol{y}}(\boldsymbol{H}\boldsymbol{x}^{(t)}) + \sqrt{2\delta}\boldsymbol{z}^{(t+1)} \\ &\quad + \frac{\alpha\delta}{\epsilon}\big(D_\epsilon(\boldsymbol{x}^{(t)}) - \boldsymbol{x}^{(t)}\big) + \frac{\delta}{\lambda}\big(\Pi_{\mathcal{C}}(\boldsymbol{x}^{(t)}) - \boldsymbol{x}^{(t)}\big),\end{aligned} \tag{3}$$

where $\boldsymbol{z}^{(t+1)} \in \mathbb{R}^N$ is a realization of a standard multivariate Gaussian distribution, and D_ϵ is a Gaussian denoiser with target noise variance $\epsilon > 0$ such that $(D_\epsilon - \mathbf{I}_N)$ is L_2-Lipschitz continuous. The parameter $\delta > 0$ is a step-size, $\alpha > 0$ is a regularisation parameter, $\lambda > 0$ is a tail regularisation parameter, and $\Pi_{\mathcal{C}}$ is the projection onto a non-empty compact convex set $\mathcal{C} \subset \mathbb{R}^N$. In imaging applications, $\mathcal{C}$ is typically a Cartesian product $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_N$.

As N and M increase, implementing (3) on a single worker becomes challenging due to hardware constraints and memory limitations. This is all the more true as building estimators and credibility intervals, necessary to uncertainty quantification, may require thousands of samples or more. Decomposing the variables $\boldsymbol{x}$ and $\boldsymbol{y}$ into partitions $(\boldsymbol{x}_b)_{1\leq b\leq B}$ and $(\boldsymbol{y}_b)_{1\leq b\leq B}$ would allow each block $\boldsymbol{x}_b$ to be sampled in parallel on different B workers, hence minimizing size-related challenges.

However, while the additive separability property of $\phi_{\boldsymbol{y}}$ implies that conditionally-independent blocks of $\boldsymbol{y}$ can be formed, a similar partitioning of $\boldsymbol{x}$ is unlikely. The problem would otherwise be trivially separable and each block $\boldsymbol{x}_b$ could be processed independently. In practice, the update $\boldsymbol{x}_b^{(t+1)}$ does not only require its past state $\boldsymbol{x}_b^{(t)}$ but also a subset of related blocks $\boldsymbol{x}_{b'}^{(t)}$, $b' \neq b$. Therefore, at each iteration, each worker b must receive from the others these extra elements of $\boldsymbol{x}^{(t)}$ to compute $\boldsymbol{x}_b^{(t+1)}$, as illustrated in fig. 1.

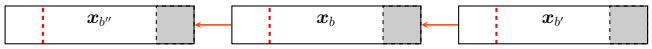


Fig. 1: Representative communication pattern for a convolution on an SPMD architecture. The subset of $\boldsymbol{x}_{b'}^{(t)}$ in dashed red lines is sent from worker b' to b, so that $\boldsymbol{x}_b^{(t)}$ can be updated by worker b using the entries received in the shaded region.

Communications are typically required in (3) for the computation of the denoising term D_ϵ and the gradient of the log-likelihood, involving $\boldsymbol{H}$ and its adjoint $\boldsymbol{H}^*$. When $\mathcal{C} = \bigtimes_{n=1}^N \mathcal{C}_n$, the projection term boils down to an element-wise projection that can be applied to the different blocks independently, defining $\mathcal{C}_b$ as a Cartesian product over the pixel indices n covered by $\boldsymbol{x}_b$. The amount of communications are the main potential bottleneck of this method. If the coupling between the blocks is too strong, which occurs when dealing with dense operators $\boldsymbol{H}$, most of the runtime will be spent in communications rather than actual computations. Communications have to be limited to obtain an efficient distributed algorithm. The good news is that many operators such as convolutions or gradients that are common in image processing are *localized*. The adjective *localized* here means that each element in the output array can be computed from a few neighbouring elements in the input array $\boldsymbol{x}$.

B. DDFB as a distributable prior

The notion of locality mentioned earlier must apply not only to $\phi_{\boldsymbol{y}}$ and $\boldsymbol{H}$ in the likelihood, but also to the operators encoding the denoiser plugged in PnP-ULA. State-of-the-art denoisers such as diffusion models [18] may offer great performance but in general do not have a structure compatible with this distributed scheme since they leverage non-local operators (e.g. attention layers). A possible alternative is to rely on convolutional networks, based on a succession of localized convolutions and element-wise non-linear operations.

State-of-the-art convolutional networks such as DRUNet allow for distributed implementation, but their size and complexity result in prohibitive communication overhead. Defined by much fewer parameters and a simpler architecture, the DDFB unfolded network [11] appears as a better alternative. It is composed of K layers, involving only convolutions and element-wise soft-thresholding operations, which can be distributed easily on SPMD architectures.

C. Proposed Distributed PnP-ULA

The very general methodology and principles above are illustrated on image deconvolution problems under additive Gaussian noise. The data fidelity term is

$$\phi_{\boldsymbol{y}}(\boldsymbol{H}\boldsymbol{x}) = \frac{1}{\sigma^2}\|\boldsymbol{H}\boldsymbol{x} - \boldsymbol{y}\|_2^2, \tag{4}$$

Algorithm 1: Proposed distributed PnP-ULA

Input: $\boldsymbol{y} \in \mathbb{R}^M, \delta \in \left(0, \frac{1}{3}(\alpha L_2/\epsilon + L_1 + 1/\lambda)^{-1}\right)$, $\lambda \in \left(0, (2L_2/\epsilon + 4L_1)^{-1}\right), \epsilon > 0, \alpha > 0$

for *each worker* $b \in \{1, \dots, B\}$ **do** in parallel

  Load and store observation chunk $\boldsymbol{y}_b \in \mathbb{R}^{M_b}$;

  Initialize local image chunk $\boldsymbol{x}_b^{(0)} \in \mathbb{R}^{N_b}$;

  for $t = 0$ **to** $T-1$ **do**

    Communicate to compute $\boldsymbol{g}_b{}^{(t)} = \frac{1}{\sigma^2}\left[\boldsymbol{H}^*\left(\boldsymbol{H}\boldsymbol{x}^{(t)} - \boldsymbol{y}\right)\right]_b$;

    Communicate for each DDFB layer to compute $\boldsymbol{d}_b{}^{(t)} = \left[D_\epsilon(\boldsymbol{x}^{(t)})\right]_b$;

    `// Local PnP-ULA transition`

    Sample $\boldsymbol{z}_b{}^{(t+1)} \sim \mathcal{N}(\boldsymbol{0}, \mathbf{I}_{N_b})$;

    $\boldsymbol{x}_b{}^{(t+1)} = \boldsymbol{x}_b{}^{(t)} - \delta \boldsymbol{g}_b{}^{(t)} + \frac{\alpha\delta}{\epsilon}\left(\boldsymbol{d}_b{}^{(t)} - \boldsymbol{x}_b{}^{(t)}\right)$

    $+\frac{\delta}{\lambda}\left(\Pi_{\mathcal{C}_b}(\boldsymbol{x}_b{}^{(t)}) - \boldsymbol{x}_b{}^{(t)}\right) + \sqrt{2\delta}\boldsymbol{z}_b{}^{(t+1)}$;

Output: $(\boldsymbol{x}^{(t)})_{1 \le t \le T}$

where $\boldsymbol{H}$ is a linear convolution operator associated with an arbitrary kernel, and $\sigma^2 > 0$ is the noise variance. The kernel size is assumed small compared to N. The function $\phi_{\boldsymbol{y}}$ is additively separable, and $\phi_{\boldsymbol{y}} \circ \boldsymbol{H}$ is L_1-Lipschitz differentiable with $L_1 = \frac{1}{\sigma^2}\|\boldsymbol{H}\|_2^2$. A pre-trained DDFB denoiser is used within the proposed distributed algorithm detailed in Algorithm 1.

This application satisfies all the requirements and assumptions from the previous paragraphs. The white noise doesn't imply any coupling between the elements of $\boldsymbol{x}$, since all noise realizations are independent. The gradient of $\phi_{\boldsymbol{y}}(\boldsymbol{H}\cdot)$ however remains non-trivially separable due to the convolution operators $\boldsymbol{H}$ and its adjoint $\boldsymbol{H}^*$. It can still be distributed efficiently as in [17] , with communication costs increasing with the size of the blurring kernel.

To compute $\boldsymbol{H}\boldsymbol{x}^{(t)}$ in Algorithm 1 algorithm 1, each worker needs to communicate border slices of its local image block $\boldsymbol{x}_b^{(t)}$ to neighboring workers. The size of these slices is conditioned by the width of the convolution kernel, enabling each worker to perform computations as if working on the entire image $\boldsymbol{x}^{(t)}$. In a similar manner, denoting $\boldsymbol{z}^{(t)} = \boldsymbol{H}\boldsymbol{x}^{(t)}$, the computation of $\boldsymbol{g}_b{}^{(t)}$ requires each worker to communicate border slices of $\boldsymbol{z}_b{}^{(t)}$ to neighbours, since $\boldsymbol{H}^*$ is also a convolution.

As DDFB only involves element-wise operations and convolutions, the same methodology can be used to compute the output of each DDFB layer. The computation of $\boldsymbol{d}_b{}^{(t)}$ in algorithm 1 can thus be carried out in a distributed manner as well.

III. Experiments on synthetic data

A. Experimental setting

The experiments have been conducted on the Jean Zay supercomputer, a CNRS facility hosted by GENCI at IDRIS. They have all been run on a single node, equipped with two 2.5 GHz 20-core Intel Cascade Lake 6248 processors, and four 16 GB-memory Nvidia Tesla V100 SXM2 GPUs. To leverage multiple GPUs efficiently, the proposed Python implementation relies on `mpi4py` for MPI-based parallelization [19], `cupy` for GPU computing [20] and `pytorch` to leverage pre-trained denoisers [21]. Codes to reproduce the experiments will be released online[1].

The DDFB network used in the proposed algorithm was defined with 19 layers, and trained with the procedure detailed in [11, Paragraph IV. A. 4), training setting 2]. All the PnP-ULA parameters were chosen following the recommendations from [14]: $\alpha = 1, \mathcal{C} = [-1, 2]^N$, $\lambda = 0.99(2L_2/\epsilon + 4L_1)^{-1}$ and $\delta = \frac{0.99}{3}(\alpha L_2/\epsilon + L_1 + 1/\lambda)^{-1}$. An approximation of L_2 was found with a power method as in [11, Section 4.4], leading to the conservative choice $L_2 = 1$, and $\epsilon = \sigma^2$.

Performance has been assessed in the following sets of experiments, based on synthetic observations generated with a fixed $\sigma = 0.03$ and motion blur kernel $\boldsymbol{H}$ of size $(27, 27)$.

a) Restoration quality: The MMSE estimate $\widehat{x}_{\text{MMSE}}$ formed from Algorithm 1 using $B \ge 1$ GPUs has been compared to the maximum *a posteriori* (MAP) estimate $\widehat{x}_{\text{MAP}}$ obtained from a PnP-Forward-Backward (PnP-FB) algorithm [22] based on DRUNet [12]. Reconstruction quality is evaluated in terms of the Structural Similarity Index Measure (SSIM) [23] and the restoration signal-to-noise ratio (rSNR)

$$\text{rSNR}(\widehat{\boldsymbol{x}}) = 10\log_{10}\left(\frac{\|\overline{\boldsymbol{x}}\|_2^2}{\|\widehat{\boldsymbol{x}} - \overline{\boldsymbol{x}}\|_2^2}\right), \tag{5}$$

with $\widehat{\boldsymbol{x}}$ an estimate of the ground truth image $\overline{\boldsymbol{x}}$. The MMSE and per pixel variances have been computed from 10^4 samples, with 10^3 burn-in samples. Credibility intervals have been computed on batches of 500 samples due to hardware memory constraints. Codes for DRUNet and corresponding weights have been retrieved from https://github.com/cszn/DPIR.

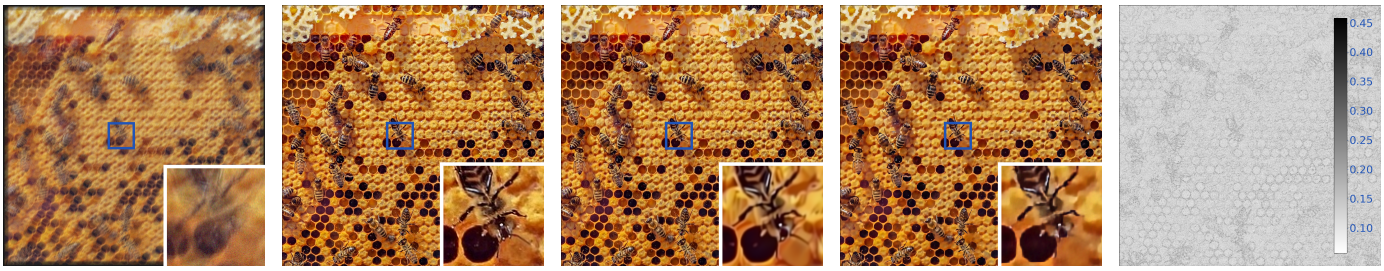
b) Scalability: The strong and weak scaling performance of the proposed distributed implementation were evaluated with one CPU core allocated to each of the $B \ge 1$ GPUs. Strong scalability is assessed by varying the number of GPUs $B \in \{1, 2, 4\}$ with a fixed image size $N = 1024 \times 1024$. In contrast, weak scaling evaluates settings with increasing image sizes and computational resources that scale proportionally, considering $(N, B) \in (1024^2, 1), (1448^2, 2), (2048^2, 4)$. The average runtime per iteration of the slowest worker has been reported as a representative time complexity metric. For $B > 1$, the images $\boldsymbol{x}$ and $\boldsymbol{y}$ were divided equally along their slower-access axis[2] for optimized memory access. Tessellations across the two spatial axes, though possible in the proposed implementation, have not been considered.

B. Results and discussion

a) Reconstruction quality: Figure 2 visually compares the proposed approach with the PnP-FB algorithm based on DRUNet. The MMSE estimate in fig. 2d was obtained using

[1] https://gitlab.cristal.univ-lille.fr/phd_maxime_bouton/dpnp-ula-ssp-2025

[2] Refers to the dimension along which data access is potentially the least contiguous in memory, hence a lower accessing efficiency.



(a) Observations $\boldsymbol{y}$ (b) Ground truth $\overline{x}$ (c) $\widehat{x}_{\text{MAP}}$ (PnP-FB) (d) $\widehat{x}_{\text{MMSE}}$ (proposed) (e) 95% CI (proposed)

Fig. 2: Restoration comparison: from left to right, the observations, the ground truth, MAP estimated with PnP-FB based on DRUNet, MMSE estimated with the proposed algorithm and $B = 4$ workers, 95% CI of the red channel of the image.

$B = 4$ workers. The zoom located on a partitioning line where communications take place confirms that no artifacts were generated due to the distributed implementation.

Table I gathers quantitative restoration metrics corresponding to the sampling of the very same problem, using 1, 2 or 4 workers for further validation. Up to negligible discrepancies due to differences in the pseudo-random number sequences, the estimates are similar: around 19.37 for the rSNR and 0.77 for the SSIM. Figure 2c shows that the MAP estimate obtained with PnP-FB using the larger DRUNet instead of DDFB yields a comparable estimate: rSNR = 19.80 and SSIM = 0.81. This experiment is meant to assess the restoration quality of the distributed algorithm, constrained by the use of a structure-specific neural network. The comparison is against a high-performance optimization algorithm employing a much larger and more intricate neural network. The ability to offer uncertainty quantification on very large problems using a distributed implementation comes at the expense of a smaller and less complex deep prior architecture. In spite of leveraging a neural network approximately 1000 times smaller than DRUNet, the proposed algorithm achieves performance that remains close, demonstrating its efficiency and effectiveness.

b) Scalability: Table I gathers the results of the strong scaling experiments. It shows that sampling was accelerated by 32% when doubling the computational resources, and by 46% with $4\times$ more resources. However, the speedup gain diminishes rapidly as the number of GPUs increases, resulting in sublinear scaling. This is primarily due to the decrease in computational benefit when processing small data chunks on each worker, ultimately failing to offset the communication costs which become dominant. There is little interest in distributing problems below a certain size. However, note that, for a fixed problem size, this behaviour highly depends on the neural network used and on the cost of required communications. The 19 layers in DDFB calls for $19 \times 2 = 38$ communication steps. Empirical observations suggest that considering a DDFB trained with twice fewer layers can divide communication costs by 2 without drastically reducing the estimation quality.

Table II illustrates the scalability potential of the proposed approach: working with a problem 4 times larger than the reference has only taken up to 46% longer [3]. The 2048×2048 imaging experiment required more than a single GPU to be run under in same settings as for the smaller problems. The efficiency of the distributed strategy decreases as the configuration grows due to larger communication costs: they scale proportionally in size and are much slower than processing.

TABLE I: Strong scaling experiment: reconstruction quality, time per iteration (in ms) and speedup for $N = 1024 \times 1024$.

Number of GPUs	Time (ms)	Speedup	rSNR	SSIM
1	199 ± 1.43	1	19.36	0.77
2	135 ± 0.87	1.47	19.37	0.77
4	108 ± 0.68	1.84	19.39	0.77

TABLE II: Weak scaling experiment

Number of GPUs (image size N)	Time (ms)	Efficiency
1 (1024×1024)	199 ± 1.43	1
2 (1448×1448)	240 ± 0.93	0.83
4 (2048×2048)	290 ± 49.4	0.69

IV. Conclusion

This article proposes a distributed algorithm that is equivalent to its sequential implementation. It leverages multiple GPUs to handle much larger problems without compromising on accuracy of the estimates. Scalability is made possible by the choice of a model suitable for distributed computations, including a deep prior with limited communications. The choice of the denoiser is a key factor. Its structure must be as light and simple as possible to limit the communications. The trade-off between the complexity of the prior and the achievable scaling efficiency is crucial. Notably, the proposed method successfully solved a large 1024×1024 image deconvolution sampling problem in 18 minutes with 4 GPUs, against 33 minutes using only 1. Finally, it enabled sampling 2048×2048 images, which would have been impossible with a sequential implementation under the same settings, showcasing its potential for high-resolution applications.

[3]The high standard deviation observed for the last timing value was due to an inconsistent external activity on the computing node during this experiment.

REFERENCES

[1] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: nonlinear phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[2] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic press, 2008.

[3] A. Chambolle and T. Pock, "An introduction to continuous optimization for imaging," *Acta Numerica*, vol. 25, pp. 161–319, 2016.

[4] M. Benning and M. Burger, "Modern regularization methods for inverse problems," *Acta numerica*, vol. 27, pp. 1–111, 2018.

[5] A. Brifman, Y. Romano, and M. Elad, "Turning a denoiser into a super-resolver using plug and play priors," in *IEEE International Conference on Image Processing*. IEEE, 2016, pp. 1404–1408.

[6] E. Ryu *et al.*, "Plug-and-play methods provably converge with properly trained denoisers," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5546–5557.

[7] U. S. Kamilov, C. A. Bouman, G. T. Buzzard, and B. Wohlberg, "Plug-and-Play Methods for Integrating Physical and Learned Models in Computational Imaging: Theory, algorithms, and applications," *IEEE Signal Process. Mag.*, vol. 40, no. 1, pp. 85–97, Jan. 2023.

[8] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, Mar. 2021.

[9] H. T. V. Le, N. Pustelnik, and M. Foare, "The faster proximal algorithm, the better unfolded deep learning architecture ? the study case of image denoising," in *European Signal Processing Conference*. IEEE, 2022, pp. 947–951.

[10] A. Repetti, M. Terris, Y. Wiaux, and J.-C. Pesquet, "Dual forward-backward unfolded network for flexible plug-and-play," in *European Signal Processing Conference*. IEEE, 2022, pp. 957–961.

[11] H. T. V. Le, A. Repetti, and N. Pustelnik, "Unfolded Proximal Neural Networks for Robust Image Gaussian Denoising," *IEEE Trans. Image Process.*, vol. 33, pp. 4475–4487, 2024.

[12] K. Zhang *et al.*, "Plug-and-Play Image Restoration with Deep Denoiser Prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6360–6376, 2022.

[13] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-Play ADMM for Image Restoration: Fixed-Point Convergence and Applications," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 84–98, Mar. 2017.

[14] R. Laumont *et al.*, "Bayesian imaging using Plug & Play priors: When Langevin meets Tweedie," *SIAM Journal on Imaging Sciences*, vol. 15, no. 2, pp. 701–737, 2022.

[15] F. Coeurdoux, N. Dobigeon, and P. Chainais, "Plug-and-Play Split Gibbs Sampler: Embedding Deep Generative Priors in Bayesian Inference," *IEEE Transactions on Image Processing*, vol. 33, pp. 3496–3507, 2024.

[16] F. Darema, "The SPMD model: Past, present and future," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Y. Cotronis and J. Dongarra, Eds., Berlin, Heidelberg, 2001, pp. 1–1.

[17] P.-A. Thouvenin, A. Repetti, and P. Chainais, "A distributed split-Gibbs sampler with hypergraph structure for high-dimensional inverse problems," *J. Comput. and Graph. Stat.*, vol. 33, no. 3, pp. 814–832, Oct. 2024.

[18] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851.

[19] L. Dalcin and Y.-L. L. Fang, "mpi4py: Status update after 12 years of development," *IEEE Comput. Sci. Eng.*, vol. 23, no. 4, pp. 47–54, 2021.

[20] R. Okuta *et al.*, "CuPy: A NumPy-compatible library for NVIDIA GPU calculations," in *Adv. in Neural Information Processing Systems*, 2017.

[21] J. Ansel *et al.*, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '24)*, vol. 2. ACM, Apr. 2024.

[22] M. Kowalski, B. Malézieux, T. Moreau, and A. Repetti, "Analysis and Synthesis Denoisers for Forward-Backward Plug-and-Play Algorithms," *ArXiv preprint*, no. arXiv:2411.13276, Dec. 2024.

[23] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.