

A Non-Monotone Preconditioned Trust-Region Method for Neural Network Training

Andrea Angino^[0009-0000-8525-375X],
Bindi Çapriqi^[0009-0001-1747-9316],
Shega Likaj^[0009-0005-8434-4672],
Ken Trotti^[0000-0002-5496-9445],
Rolf Krause^[0000-0001-5408-5271]

1 Introduction

Given samples $\{(x^{(i)}, y^{(i)})\}_{i=1}^M$ with $x^{(i)} \in \mathbb{R}^q$ and $y^{(i)} \in \mathbb{R}^p$, we train a neural network (NN) by solving for $\theta \in \mathbb{R}^n$

$$\min_{\theta \in \mathbb{R}^n} f(\theta) = \frac{1}{M} \sum_{i=1}^M \mathcal{L}(\mathcal{N}(\theta; x^{(i)}), y^{(i)}),$$

where $\mathcal{L} : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}$ is a per-sample loss and $\mathcal{N}(\theta; \cdot)$ maps inputs to predictions. In modern applications, f is large-scale, nonconvex, and typically evaluated under sampling noise.

First-order methods such as stochastic gradient descent (SGD) and Adam remain the default [8, 10], with performance largely governed by step-size and momentum schedules. This leads to many hyper-parameters that must be tuned and does not exploit the pipelined execution of the model across multiple devices, where a NN is split into sequential blocks placed on different GPUs and activations are forwarded

Andrea Angino
UniDistance Suisse, Switzerland, e-mail: andrea.angino@unidistance.ch

Bindi Çapriqi
King Abdullah University of Science and Technology, Saudi Arabia, e-mail:
bindi.capriqi@kaust.edu.sa

Shega Likaj
Università della Svizzera italiana, Switzerland & King Abdullah University of Science and
Technology, Saudi Arabia, e-mail: shega.likaj@kaust.edu.sa

Ken Trotti
King Abdullah University of Science and Technology, Saudi Arabia, e-mail: ken.trotti@kaust.edu.sa

Rolf Krause
Università della Svizzera italiana, Switzerland & King Abdullah University of Science and
Technology, Saudi Arabia, e-mail: rolf.krause@kaust.edu.sa

from one device to the next. These issues motivate the study of algorithms that explicitly exploit such multi-device pipelines, and rely on globalization strategies which adapt parameters (e.g., step sizes) automatically.

Domain decomposition (DD) enables model-parallelism by partitioning parameters into subdomains and updating restricted subproblems in parallel with additive recombination [2, 14, 7]. This mirrors classical DD for PDEs [14] and aligns with modern model-parallel pipelines [1, 12]. For globalization, trust-region (TR) strategies provide robust control for nonconvex problems [13, 3] and have been adapted to multi-level and mini-batch regimes [6, 11]. Within DD, additive updates with a TR safeguard underlie the Additively Preconditioned Trust-Region Strategy (APTS) [9].

In this paper, we extend the APTS framework [4, 9, 5, 15] by incorporating the principles of non-monotone TR (NTR) methods, resulting in the Non-monotone Additively Preconditioned Trust-Region method (NAPTS). NAPTS compares trial steps against a sliding reference value, which relaxes strict monotonic decrease, accommodates noisy mini-batch objectives, and enables more persistent coarse-space directions.

2 Foundations of NAPTS

This section describes the subdomains structure, the local solves, and the globalization strategy.

2.1 Additive domain decomposition update

Let $\{C_d\}_{d=1}^N$ be a partition of $\{1, \dots, n\}$ with restriction and prolongation operators $R_d : \mathbb{R}^n \rightarrow \mathbb{R}^{n_d}$ and $R_d^T : \mathbb{R}^{n_d} \rightarrow \mathbb{R}^n$ satisfying $R_d R_d^T = I_{n_d}$, $R_d R_{d'}^T = 0$ for $d \neq d'$, and $\sum_{d=1}^N R_d^T R_d = I_n$. This is the non overlapping parameter space analogue of an additive Schwarz decomposition, where the “domain” is the NN parameter space.

To make this concrete, Figure 1a shows an example of such a decomposition with $N = 3$ for a fully connected NN, where each color identifies one subdomain of the parameter partitions C_d .

To let the subdomains operate independently, we first expose them to global information. Therefore, given a global iterate θ^k at iteration k of NAPTS, we compute the objective value $f(\theta^k)$ and its gradient $\nabla f(\theta^k)$ by running a single forward and backward propagation through the full NN, during which we cache the activation values and downstream gradients (black/red arrows in Figure 1b). To illustrate this idea for the simple case of the NN in Figure 1, let us define the subdomains D_d as:

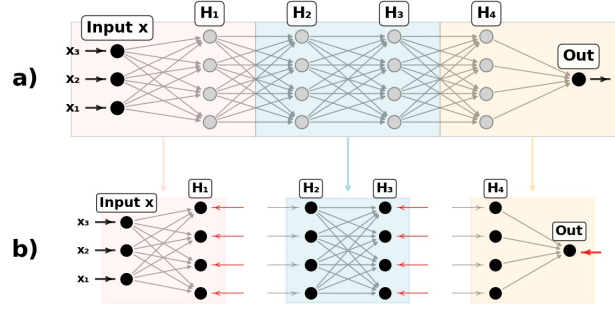


Fig. 1 A NN decomposition example. (a) Hardware-based partition into three blocks. (b) Black/red arrows denote cached boundary activations/downstream gradients.

$$\begin{aligned}
 D_1(x; \theta_1) &= \sigma_1(W_{H_1}x + b_{H_1}), \\
 \text{Subdomains: } D_2(x; \theta_2) &= \sigma_2(W_{H_2} \sigma_3(W_{H_3}x + b_{H_3}) + b_{H_2}), \\
 D_3(x; \theta_3) &= \sigma_{\text{out}}(W_{\text{out}} \sigma_4(W_{H_4}x + b_{H_4}) + b_{\text{out}}), \\
 \text{Full NN: } \mathcal{N}(x; \theta) &= D_3(D_2(D_1(x))),
 \end{aligned}$$

where σ_i are the activation functions and θ contains all the network parameters $\theta_d = R_d \theta \forall d$, which split into weights W and biases b . By the chain rule and linearity, the gradient $\nabla f(\theta)$ can be factored into a downstream gradient and a local derivative, as summarized in Table 1 where \hat{y} denotes the NN output on the full dataset or current minibatch.

Parameter θ_d	Downstream gradient G_d	Gradient $\frac{\partial f}{\partial \theta_d}$
θ_3	$G_3 = \frac{\partial f}{\partial \hat{y}} = \frac{\partial f}{\partial D_3}$	$\frac{\partial f}{\partial \theta_3} = G_3 \frac{\partial D_3}{\partial \theta_3}$
θ_2	$G_2 = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial D_2} = \frac{\partial f}{\partial D_2}$	$\frac{\partial f}{\partial \theta_2} = G_2 \frac{\partial D_2}{\partial \theta_2}$
θ_1	$G_1 = \frac{\partial f}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial D_2} \frac{\partial D_2}{\partial D_1} = \frac{\partial f}{\partial D_1}$	$\frac{\partial f}{\partial \theta_1} = G_1 \frac{\partial D_1}{\partial \theta_1}$

Table 1 Stored downstream and blockwise gradients for each θ_d .

Thus, during local iterations, the cached forward activation from the previous subdomain provides the input needed to evaluate the local derivative $\partial D_d / \partial \theta_d$, while the stored backward gradient from the next subdomain, $G_{d+1} = \partial f / \partial D_{d+1}$, allows us to reconstruct G_d . Starting from $\theta_d^{k,0} = R_d \theta^k$, during the ℓ inner iterations $t = 1, \dots, \ell$, we keep the stored downstream gradient G_d^k fixed, while recomputing the local derivative $\partial D_d / \partial \theta_d$ using cached boundary activations. This yields approximate block-gradients of the form

$$\tilde{g}_d^{k,t} := G_d^k \frac{\partial D_d}{\partial \theta_d}(x_d^k; \theta_d^{k,t}) \approx \frac{\partial f}{\partial \theta_d}(\theta^{k,t}), \quad t = 1, \dots, \ell, \quad (1)$$

where x_d^k denotes the cached input of subdomain d at the iteration k .

For the local subdomain updates, we use an Adam-type method where the gradients are computed through equation (1) and Adam step is truncated to satisfy the per-step bound $\Delta_d^k := \Delta^k / \ell$, i.e.,

$$s_d^{k,t} = \min \left\{ \left\| \tilde{s}_d^{k,t} \right\|_\infty, \Delta_d^k \right\} \frac{\tilde{s}_d^{k,t}}{\left\| \tilde{s}_d^{k,t} \right\|_\infty} \quad \text{and update} \quad \theta_d^{k,t} = \theta_d^{k,t-1} + s_d^{k,t}. \quad (2)$$

Therefore, after ℓ local iterations, the subdomain d yields $s_d^k = \sum_{t=1}^{\ell} s_d^{k,t}$, such that $\|s_d^k\|_\infty \leq \Delta_d^k$. The global trial step is the additive lift of the N local updates $\{s_d^k\}_{d=1}^N$,

$$s^k = \sum_{d=1}^N R_d^T s_d^k. \quad (3)$$

Finally, we note that communications are not needed during the subdomain training and that this construction is not tied to the NN structure in Figure 1 and generalizes to more complex architectures.

2.2 Non-monotone trust-region as globalization strategy

In this section, we follow the NTR framework [3, Alg. 10.1.1]. Given the proposed additive search direction s^k in (3), the NTR mechanism acts, in NAPTS, as a globalization strategy deciding whether to accept the global step s^k . In contrast to previous APTS variants [4, 5, 15] with exact gradient and based on monotone TR schemes, we deliberately switch to a non-monotone variant to increase the likelihood of accepting coarse space steps, and thereby avoid expensive recomputation of the search direction. This is particularly appropriate in our setting, where the coarse-space updates are inexact and potentially batch-based, and are therefore inherently non-monotone. Preliminary experiments in [15] suggest that aggressively accepting coarse-space directions can accelerate loss decrease even at the price of temporary objective increases, but this comes at the cost of an indiscriminate acceptance policy. In NAPTS, we therefore introduce a more robust NTR mechanism that selectively accepts non-monotone coarse-space steps in a controlled way.

We introduce a local model m^k that approximates f in the global trust-region

$$\mathcal{B}_G^k := \{s \in \mathbb{R}^n : \|s\|_\infty \leq \Delta^k\}, \quad \Delta^k > 0.$$

Classically, TR methods employ a quadratic model and define the trial step as an approximate solution of the corresponding trust-region subproblem over \mathcal{B}_G^k .

However, in our large-scale setting, forming and manipulating exact second-order information is prohibitive, and using approximate second-order terms introduces additional complications. For simplicity, we therefore focus on a first-order model

$$m^k(s) = f(\theta^k) + \nabla f(\theta^k)^\top s,$$

and use step s^k in equation (3) as trial step that satisfies $\|s^k\|_\infty \leq \Delta^k$.

Let $\nu \in \mathbb{N}$ be the memory parameter and let \mathcal{W}_ν^k be the set that keeps track of the indices of the most recent ν successful iterations, i.e.

$$\mathcal{W}_\nu^k := \{j \in \{\max\{0, k - \nu\}, \dots, k\} \mid \text{iteration } j \text{ is successful}\}. \quad (4)$$

We then define the reference-index map $r : \mathbb{N} \rightarrow \mathbb{N}$ as

$$r(k) := \arg \max_{j \in \mathcal{W}_\nu^k} f(\theta^j),$$

so that $r(k) \in \mathcal{W}_\nu^k$. The corresponding history term $\sigma_h^k \geq 0$ is defined as the cumulative predicted expected decrease among successful iterations between the reference index $r(k)$ and the current one,

$$\sigma_h^k := \sum_{\substack{i=r(k) \\ i \in \mathcal{W}_\nu^k}}^{k-1} [m^i(0) - m^i(s^i)]. \quad (5)$$

Once s^k is computed, the quality of the model prediction is assessed via two agreement ratios

$$\rho_c^k = \frac{f(\theta^k) - f(\theta^k + s^k)}{m^k(0) - m^k(s^k)}, \quad \rho_h^k = \frac{f(\theta^{r(k)}) - f(\theta^k + s^k)}{\sigma_h^k + m^k(0) - m^k(s^k)}, \quad (6)$$

where the subscripts c and h refer to *current* and *historical* quantities, respectively. Specifically, we set $\rho^k := \max\{\rho_c^k, \rho_h^k\}$ and use this combined ratio in place of the classical TR ratio ρ_c^k to decide step acceptance and to update the global radius Δ^k .

By construction, $f(\theta^{r(k)}) \geq f(\theta^k)$, so ρ_h^k may be larger than ρ_c^k and accept the step even in the case where $f(\theta^k + s^k) > f(\theta^k)$. Hence, the sequence $\{f(\theta^k)\}_k$ is not required to be monotonically decreasing, while the reference values $\{f(\theta^{r(k)})\}_k$ and the history term σ_h^k still enforce a long-term descent behaviour. The additional overhead with respect to a standard monotone TR scheme is negligible: maintaining the window \mathcal{W}_ν^k , the index $r(k)$, and the scalar σ_h^k requires only simple updates per iteration, yet this non-monotone mechanism can substantially increase the number of accepted steps coming from coarse spaces, thereby reducing the need for expensive global corrections of the search direction.

2.3 The NAPTS Method

The NAPTS method, summarized in Algorithm 1, combines parallel subdomain updates with two NTR stages. Each iteration naturally decomposes into two phases: parallel subdomain computations, preconditioning, and NTR steps. In the first phase, lines 4–6, each subdomain D_d performs in parallel ℓ local constrained Adam(CAdam) steps. This produces local steps s_d^k which are lifted and combined to form the global proposal s^k as in (3).

The second phase (lines 7–9) applies the NTR update strategy to the obtained proposal s^k . Using the reference quantities \mathcal{W}_v^k , $r(k)$, and σ_h^k , we compute the current and historical agreement ratios ρ_c^k and ρ_h^k and combine them into $\rho^k = \max\{\rho_c^k, \rho_h^k\}$ (cf. (6)). ρ^k is then used to decide whether to take the step s^k or a corrected step c^k , and to update the global radius Δ^k . The correction c^k in line 8 is a convex combination of a steepest descent step of length Δ^k and the global proposal s^k , for some $\alpha_k, \beta_k \in [0, 1]$, namely,

$$c^k = \beta_k \left[(1 - \alpha_k) \left(-\Delta^k \frac{\nabla f(\theta^k)}{\|\nabla f(\theta^k)\|} \right) + \alpha_k s^k \right]. \quad (7)$$

Coefficients $(\alpha, \beta) \in \{(0.8, \frac{1}{2}), (0.6, \frac{1}{4}), (0.4, \frac{1}{8}), (0.2, \frac{1}{16}), (0, \frac{1}{32})\}$ are tested through a for loop. This construction avoids wasting the costly subspace updates and gradient computations when the proposed step s^k would otherwise be rejected.

In the third phase at line 10, a classical NTR step is performed, as described in Section 2.2, starting from $(\theta^{k+\frac{1}{2}}, \Delta^{k+\frac{1}{2}})$ and using a search direction based on a first-order model. We note that setting $\nu = 1$ recovers the APTS method studied in [15] with approximated subdomains.

3 Numerical examples

We consider image classification on the CIFAR-10 dataset. For this task, we use a convolutional NN (CNN¹) with four convolutional blocks and two fully-connected layers (1.2M parameters) trained in a multi-GPU distributed setting on a compute node with four Nvidia A100 GPUs using data mini-batches of size 1,000. The model decomposition is layer-based, with each subdomain corresponding to a contiguous block of layers, consistent with the schematic illustrated in Figure 1.

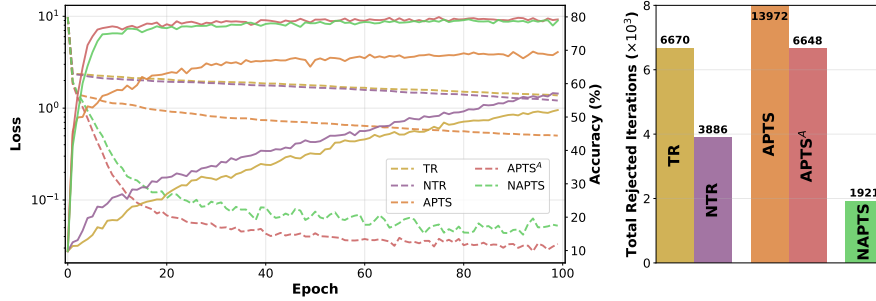
A rejection is counted whenever a search direction fails a TR acceptance test, either at the globalization level or within the internal correction loop used to construct c^k .

Left panel of Figure 2 reports the training loss and validation accuracy, and the right panel the rejection counts per batch, for TR (53.65s/epoch), NTR (47.26s/epoch), APTS (169.14s/epoch), APTS^A (152.39s/epoch), and the proposed method NAPTS (115.88s/epoch). Training loss is an optimization diagnostic and

¹ A CNN is a feedforward architecture that applies learned convolutional filters; see [8, Ch. 9].

Algorithm 1 NAPTS: Non-monotone Additively Preconditioned Trust-Region Strategy**Input:** $f : \mathbb{R}^n \rightarrow \mathbb{R}$, initial iterate $\theta^0 \in \mathbb{R}^n$, restriction operators $\{R_d\}_{d=1}^N$ **Output:** Approximate minimizer θ^* of f **Constants:** inner iterations ℓ , memory ν , NTR parameters $(\eta_1, \eta_2, \gamma_{\text{dec}}, \gamma_{\text{inc}}, \Delta^0)$ \triangleright See [3]1: Set $k \leftarrow 0$ and initialize NTR history $(\sigma_h^0, r(0), \mathcal{W}_\nu^0)$ \triangleright cf. Section 2.22: **while** not converged **do**3: Compute $f(\theta^k), \nabla f(\theta^k)$, block gradients $g_d^k \leftarrow R_d \nabla f(\theta^k)$ 4: **for all** $d = 1, \dots, N$ **in parallel do** \triangleright Phase 1: parallel subdomains5: $s_d^k \leftarrow \text{CAAdam}(g_d^k, \ell, \Delta_d^k)$ \triangleright local step, cf. (2)6: Form the global proposal $s^k = \sum_{d=1}^N R_d^T s_d^k$ 7: Compute $\rho_c^k, \rho_h^k, \rho^k = \max\{\rho_c^k, \rho_h^k\}$ \triangleright Phase 2: NTR, cf. (6)8: Step update \triangleright cf. (7)

$$\theta^{k+\frac{1}{2}} := \begin{cases} \theta^k + s^k, & \text{if } \rho^k > \eta_1, \\ \theta^k + c^k, & \text{otherwise,} \end{cases} \quad \Delta^{k+\frac{1}{2}} := \begin{cases} \gamma_{\text{inc}} \Delta^k, & \text{if } \rho^k \geq \eta_2, \\ \Delta^k, & \text{if } \rho^k \in [\eta_1, \eta_2), \\ \gamma_{\text{dec}} \Delta^k, & \text{if } \rho^k < \eta_1. \end{cases}$$

9: Update $\mathcal{W}_\nu^k, r(k), \sigma_h^k$, \triangleright cf. (4)–(5)10: $\theta^{k+1}, \Delta^{k+1} \leftarrow \text{NTR}(\theta^{k+\frac{1}{2}}, \Delta^{k+\frac{1}{2}}, \nabla f(\theta^{k+\frac{1}{2}}))$ \triangleright Phase 3: Smoothing NTR iteration11: $k \leftarrow k + 1$ 12: **return** $\theta^* := \theta^k$ **Fig. 2** Loss and accuracy (left) and rejected steps per batch (right). NTR and NAPTS use $\nu = 100$, and the superscript A denotes always accepting the global step in (3).

does not, by itself, imply better predictive performance. For this reason, we assess performance primarily through the validation accuracy, i.e., the fraction of correctly classified samples on the held-out validation set.

The baseline TR and NTR schemes struggle to reach high accuracy. Although they require roughly half the backpropagations per epoch relative to the APTS-based methods, they still fall short of comparable accuracy even after twice as many epochs. Nevertheless, NTR improves over TR, with about half the rejected directions and better loss and accuracy. This gap motivates a non-monotone globalization within APTS, which is well suited to the nonconvex and noisy training landscape.

In contrast, the preconditioned variants APTS and NAPTS achieve lower training losses and substantially higher validation accuracies, suggesting improved optimization without loss of generalization. As shown in [15], the aggressive-acceptance variant APTS^A is also effective, yielding about a 10% reduction in CPU time rela-

tive to APTS by always accepting the global step in (3). Moreover, NAPTS provides a clear improvement over APTS^A by retaining a controlled loss-based acceptance mechanism that enhances robustness, while delivering a further $\approx 20\%$ reduction in CPU time compared to APTS^A (about 30% relative to APTS). This gain follows directly from the marked decrease in rejected steps visible in the right panel, indicating that the preconditioned step is almost always accepted.

In our setup, Adam/SGD requires $\approx 40\text{s/epoch}$ (about $3\times$ faster), but typically needs learning-rate and schedule tuning across multiple runs. In contrast, APTS/NAPTS uses trust regions to adapt step sizes automatically. Overall, NAPTS improves acceptance of coarse-space proposals, reducing rejections and CPU time.

Acknowledgements The research was funded by the Swiss National Science Foundation (SNSF) under projects No. 224943 and No. 197041 (ML²).

References

1. Ben-Nun, T., Hoefler, T.: Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. *ACM Comput. Surv.* **52**(1), 1–43 (2019)
2. Chan, T.F., Zou, J.: Additive Schwarz Domain Decomposition Methods for Elliptic Problems on Unstructured Meshes. *Numer. Algorithms* **8**(2), 329–346 (1994)
3. Conn, A.R., Gould, N.I., Toint, P.L.: Trust region methods. Society for Industrial and Applied Mathematics (2000)
4. Cruz Alegría, S., Çapriqi, B., Likaj, S., Trotti, K., Krause, R.: An Additively Preconditioned Trust-Region Strategy for Machine Learning. arXiv preprint arXiv:2512.14286 (2025)
5. Cruz Alegría, S., Trotti, K., Kopaničáková, A., Krause, R.: Data-parallel neural network training via nonlinearly preconditioned trust-region method. In: ENUMATH 2023, *Lect. Notes Comput. Sci. Eng.*, vol. 153, pp. 34–43. Springer, Berlin, Germany (2025)
6. Curtis, F.E., Scheinberg, K., Shi, R.: A Stochastic Trust-Region Algorithm Based on Careful Step Normalization. *INFORMS J. Optim.* **1**, 200–220 (2019)
7. Erhel, J., Gander, M.J., Halpern, L., Pichot, G., Sassi, T., Widlund, O.: Domain Decomposition Methods in Science and Engineering XXI. Springer, Switzerland (2014)
8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
9. Groß, C.: A Unifying Theory for Nonlinear Additively and Multiplicatively Preconditioned Globalization Strategies: Convergence Results and Examples From the Field of Nonlinear Elastostatics and Elastodynamics. Ph.D. thesis, Bonn International Graduate School, University of Bonn, Bonn, Germany (2009)
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, May 7–9, 2015 (2015)
11. Kopaničáková, A., Krause, R.: Globally Convergent Multilevel Training of Deep Residual Networks. *SIAM J. Sci. Comput.* **45**(3), S254–S280 (2023)
12. Nichols, D., Singh, S., Lin, S.H., Bhatele, A.: A Survey and Empirical Evaluation of Parallel Deep Learning Frameworks. arXiv preprint arXiv:2111.04949 (2021)
13. Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York, NY (1999)
14. Toselli, A., Widlund, O.: Domain Decomposition Methods: Algorithms and Theory, *Springer Ser. Comput. Math.*, vol. 34. Springer, Berlin, Germany (2004)
15. Trotti, K., Cruz Alegría, S., Krause, R., Kopaničáková, A.: Parallel trust-region approaches in neural network training. In: Proceedings of the MATH+ Thematic Einstein Semester 2023: Mathematical Optimization for Machine Learning, pp. 107–120. De Gruyter, Berlin (2025)