Peertechz

ENGINEERING GROUP

TRENDS IN
Computer Science and Information Technology
OPEN ACCESS JOURNAL

ISSN: 2641-3086 DOI: https://dx.doi.org/10.17352/tcsit

Short Communication

A Model Decomposition-in-Time of Recurrent Neural Networks: A Feasibility Analysis

L D'Amore*

University of Naples Federico II, Naples, Italy

Received: 03 February, 2025
Accepted: 18 February, 2025
Published: 19 February, 2025

***Corresponding author**: L D'Amore, University of Naples Federico II, Naples, Italy, E-mail: luisa.damore@unina.it; luisa.damore@dma.unina.it

Keywords: Parallel-in-time approach; Domain decomposition;Additiveobjectivefunction;Constrained least square problems; Recurrent neural networks; Parallel algorithm

https://www.engineegroup.us

Check for updates

Abstract

In the context of Recurrent Neural Networks, minimization of the Loss Function (LF) causes the most training overhead. Following the Parallel In-Time approaches, we introduce an ab-initio decomposition across time direction. The key point of our approach lies in the innovative definition of local objective functions which allows us to overcome the sequential nature of the network and the management of dependencies between time steps. In particular, we define local RNNs by adding a suitable overlapping operator to the local objective functions which guarantees their matching between adjacent subsequences. In this way, we get to a fully parallelizable decomposition of the RNN whose implementation avoids global synchronizations or pipelining. Nearest neighbours communications guarantee the algorithm's convergence. We hope that these findings encourage readers to further extend the framework according to their specific application requirements.

1. Introduction

Recurrent Neural Networks (RNNs) are a class of Deep Learning models that are fundamentally designed to handle sequential data. Unlike feedforward neural networks, RNNs possess the unique feature of maintaining a memory of previous inputs by using their internal state to process sequences of inputs. This makes them ideally suited for applications where the order of data points is crucial [1]. With the increase in the size of the dataset and the improvement of the complexity of the model, the demand for computational strength and storage space in the training process also increases proportionally. The present work is placed in the context of the design of parallel algorithms for RNNs in large-scale applications where parallelizing RNNs can be challenging due to their sequential nature. We focus on RNNs with an additive loss function consisting of a large number of component functions, let us say f_i, such as

$$\mathrm{L}(\theta) = min \sum_i f_i(\theta) \quad (1)$$

where each term corresponds to the error between some data and the output of a parametric model, with being the vector of parameters. An example is linear least squares problems, where f_i has a quadratic structure, except for a regularization function. A more general class of additive cost problems is nonlinear least squares [2,3].

Minimization of the Loss Function (LF) causes the most training overhead. The structure of the additive cost function has facilitated the use of parallel computing approaches that are well-suited for the incremental approach [4]. Very often, besides the exploitation of Graphics Processing Units' acceleration [5], concurrency is introduced inside the operations of each minimization step, at the cost of an all-to-one data synchronization at the end of each step (this is the simplest way to achieve what is defined in the context of deep learning as data parallelism or the mini-batch gradient descent approach). In the alternative, the incremental computing of the values and subgradients of the components f_i could be performed in a distributed manner. This is defined as model parallelism. Naive model parallelism is relatively simple. It's straightforward to split a large model into chunks of consecutive layers. However, there's a sequential dependency between inputs and outputs of layers, so a naive implementation can lead to a large amount of idle time while a worker waits for outputs from the previous machine to be used as its inputs. We can reuse the ideas from data parallelism by having each worker only process a subset

Citation: D'Amore L. A Model Decomposition-in-Time of Recurrent Neural Networks: A Feasibility Analysis. Trends Comput Sci Inf Technol. 2025;10(1):007-010. Available from: https://dx.doi.org/10.17352/tcsit.000091

of data elements at one time, allowing us to cleverly overlap new computations with wait time. Pipeline parallelism allows the execution of a model to be partitioned such that multiple micro-batches can execute different parts of the model code concurrently [6-11].

In the presence of evolutionary problems, in the last decades, Parallel-In-Time methods have been investigated for reducing the temporal dimensionality of such problems. Since Nievergelt, in 1964, proposed for the first time the decomposition algorithm for finding the parallel solutions of evolutionary ordinary differential equations [12], the methods of time-parallel time integration have been extensively expanded and several relevant works can be found in the literature. An extensive and updated literature list can be found on the website [13] collecting information about people, methods, and software in the field of parallel-in-time integration methods. By relying on results we have obtained on the Kalman Filter and on variational methods on the uncertainty quantification models [14-16], in this work we present the main idea underlying an *ab-initio* model decomposition in time of RNNs. This approach implies both the time series and the objective function decomposition among computing resources. Local functionals are suitably modified, by imposing a regularization constraint in order to enforce the matching of their solutions between adjacent sub-sequences. Finally, according to the Additive Schwarz Method (ASM), synchronization of local solutions is imposed iteratively. Such synchronization guarantees the model convergence [16].

2. Model decomposition-in-time of RNNs

In the following, we summarize the key components of the hybrid decomposition of RNNs. In particular, we introduce the operators defining data reduction and localization then, finally, we give the mathematics underlying decomposition in time of RNNs. For simplicity, we consider the simplest RNN, with a self-connected hidden state, through which information cycles across time steps [17]. The decomposition approach could be extended to other types of RNN.

2.1 Recurrent neural networks

RNNs are a class of deep learning models that are fundamentally designed to handle sequential data. At each time step, $t = 1,...,q$, the RNN takes an input vector, $x_t \in \Re^{m_0}$ and returns the output vector $y_t \in \Re^{m_1}$, s.t.

$$y_t = \sigma_y(W_{ht}h_t + b_t) \tag{2}$$

where

$$\begin{aligned} h_t &= \sigma_h(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad t > 0, \\ h_0 &= 0 \qquad\qquad\qquad\qquad\quad t = 0, \end{aligned} \tag{3}$$

is the hidden state vector that captures information about previous inputs, W_{xh} is the weight matrix between the input and hidden layer, W_{hh} is the weight matrix for the recurrent connection (hidden-to-hidden), b_t and b_h are the bias vectors, and finally σ_y and σ_h are the activation functions. The weight matrices and the bias functions characterize the unknown model. These parameters are obtained by minimizing a predefined objective Function.

For convenience, at each time $t = 1,..., q$ we concatenate vectors x_t and h_{t-1} to form a $m_0 + m_1$ dimensional vector a_t. Then, if we pose

$$W_I = (W_{xh}), \quad W_L = (W_{hh})$$

where $W_I \in \mathbb{R}^{m_1 \times m_0}, W_L \in \mathbb{R}^{m_1 \times m_1}, x_t \in \mathbb{R}^{m_0}, h_{t-1} \in \mathbb{R}^{m_1}$, we write all the weights together in the matrix

$$W = (W_I, W_L) \in \Re^{m_1 \times (m_0 + m_1)}$$

It results that equations (2) and (3) can be written as

$$y_t = \sigma_t(Wa_t) \quad t = 1,\ldots,q. \tag{4}$$

Let $G_t \in \Re^{m_1 \times (m_0 + m_1)}$ be the matrix obtained by each r.h.s. of equation in (4); $g = vec(G_t) \in \mathbb{R}^{(m_0+m_1)\times(m_1 \times q)}$ be the vector obtained by stacking the t r.h.s of the equation in (4) on top of one another; let $z \in \Re^{m_1 \times q}$ be the column vector obtained collecting the vector y_t (output of the RNN) q times. Finally, let $A \in \Re^{q \times (m_0 + m1)}$ be the matrix whose t-th row is obtained by the vectors a_t, $t = 1...,q$. The equation in (4) can be written as the overdetermined linear system [18,19].

$$Ag = z$$

(Least Square Minimization in RNNs) The least-square minimization problem refers to the computation of g such that [20]:

$$\hat{g} = argmin_g J(g) \tag{5}$$

with objective function

$$J(g) = \| Ag - z \|_2 \tag{6}$$

2.2 Model decomposition in time

The Model Decomposition in Time direction implies the reduction of both the time series and the objective function among computing resources. This is obtained by data reduction operation and functional localization, respectively. Local functions are obtained as a suitable modification of the original objective function. In particular, local functions are composed of two parts: the first part is the restriction of the original objective function to the sub-interval, then a regularization constraint is added in order to enforce the matching of their solutions between adjacent sub-sequences.

(Data Reduction) Let $w = [w_t w_{t+1} ... w_n]^T \in R^s$ be a vector with $t \geq 1$, $n > 0$, $s = n - t$ and $I_r = \{1,...,r\}$, $r > n$ and $n > t$. The extension of w to I_r is:

$$EO_{I_r} : w \in \mathbb{R}^S \rightarrow EO_{I_r}(w) = [\bar{w}_1\ \bar{w}_2 \ldots \bar{w}_r]^T \in \mathbb{R}^r, \tag{7}$$

where for $i = 1,...,r$

$$\bar{w}_i = \begin{cases} w_i & if\ t \leq i \leq n \\ 0 & if\ i > n and i < t \end{cases} \tag{8}$$

Let $B = [B^1 B^2 ... B^n] \in R^{m \times n}$ be such that B^j is the j-th column of

Citation: D'Amore L. A Model Decomposition-in-Time of Recurrent Neural Networks: A Feasibility Analysis. Trends Comput Sci Inf Technol. 2025;10(1):007-010. Available from: https://dx.doi.org/10.17352/tcsit.000091

B. For $i < j$ ($i = 1,...,n-1$ and $j = 2,...,n$) we define the set $I_{i,j} = \{i,...,j\}$. The restriction of B to the set I_j is:

$$|_{I_j} : B \in \mathbb{R}^{m\times n} \to B|_{I_j} = [B^1\ B^2 \dots B^j] \in \mathbb{R}^{m\times j}, \tag{9}$$

and to $I_{i,j}$ is:

$$|_{I_{i,j}} : B \in \mathbb{R}^{m\times n} \to B|_{I_{i,j}} = [B^i\ B^{i+1} \dots B^j] \in \mathbb{R}^{m\times j-i}. \tag{10}$$

(Objective Function Reduction) Let $n_1, n_2 > 0$ two natural numbers less than q. Let

$$J|_{(I_{n_i},I_{n_j})} : (u|_{I_{n_i}}, u|_{I_{n_j}}) \mapsto J|_{(I_{n_i},I_{n_j})}(u|_{I_{n_i}}, u|_{I_{n_j}}) \quad \forall i,j = 1,2 \tag{11}$$

denote the restriction of J defined in (6).

For simplicity of notations, we let $J_{i,j} \boxtimes J|_{(Ii,Ij)}$ with $i, j = 1,2$. We consider the decomposition of RNNs in two subsets only.

(In Time DD-RNN setup) Let $I = \{1,...,q\}$ be the index set of rows of A. DD-RNN setup consists in decomposing I into 2 sets:

$$I_{n_1} = \{1,...,n_1\}, \quad I_{n_2} = \{n_1 - s + 1,...,q\}, \tag{12}$$

where $|I_1| = n_1 > 0$, $|I_2| = n_2 > 0$, and

$$I_{1,2} = \{n_1 - s + 1,...,n_1\}, \tag{13}$$

is the overlap set. If $s = 0$, then $I_1 \boxtimes I_2 = \emptyset$ and $I_{1,2} \neq \emptyset$. Restrictions of A to I_1 and I_2 are:

$$A_1 = A|_{I_1} \in \mathbb{R}^{n_1\times(m_0+m_1)}, \quad A_2 = A|_{I_2} \in \mathbb{R}^{n_2\times(m_0+m_1)}, \tag{14}$$

(In Time DD-RNN Algorithm) According to the ASM in [21], we introduce the loop for $k = 0,1,2,...$, the model decomposition-in-time leads to two RNNs solving iteratively the following reduced least square minimization problems:

$$\begin{aligned} P_1^{k+1} : \hat{g}_1^{k+1} &= \underset{g_1^{k+1}\in\mathbb{R}^{n_1}}{argmin} J_1(g_1^{k+1}, g_2^k) \\ &= \underset{g_1^{k+1}\in\mathbb{R}^{k_1}}{argmin} \left[J|_{(I_1,I_2)}(g_1^{k+1}, g_2^k) + \mu \cdot \mathcal{O}_{1,2}(g_1^{k+1}, g_2^k) \right] \end{aligned} \tag{15}$$

$$\begin{aligned} P_2^{n+1} : \hat{g}_2^{k+1} &= \underset{g_2^{k+1}\in\mathbb{R}^{n_2}}{argmin} J_2(g_2^{k+1}, g_1^{k+1}) \\ &= \underset{g_2^{k+1}\in\mathbb{R}^{k_2}}{argmin} \left[J|_{(I_2,I_1)}(g_2^{k+1}, g_1^{k+1}) + \mu \cdot \mathcal{O}_{1,2}(g_2^{k+1}, g_1^k) \right] \end{aligned} \tag{16}$$

$O_{1,2}$ is the overlapping operator representing the data exchange on the overlap set $I_{1,2}$, and $\mu > 0$ is the regularization parameter.

In particular, we pose

$$\mathcal{O}_{1,2}(g_i, g_j) = \| EO_{I_i}(g_i|_{I_{1,2}}) - EO_{I_i}(g_j|_{I_{1,2}}) \|, i,j = 1,2 \tag{17}$$

with $EO_{I_i} g_1|_{I_{1,2}}$, $EO_{I_i} g_2|_{I_{1,2}}$ be the extension to I_i, of restriction to $I_{1,2}$ in (13) of $g_1 \in \mathbb{R}^{n_1}$ and $g_2 \in \mathbb{R}^{n_2}$, respectively.

As a solution, we get the sequence $\{g^{k+1}\}_{k\in\mathbb{N}_0}$ such that:

$$g^{k+1} = \begin{cases} \hat{g}_1^{k+1}|_{I_1\setminus I_{1,2}} & on\ I_1 \setminus I_{1,2} \\ \frac{\mu}{2}(\hat{g}_2^{k+1}|_{I_{1,2}} + \hat{g}_1^{k+1}|_{I_{1,2}}) & on\ I_{1,2} \\ \hat{g}_2^{k+1}|_{I_2\setminus I_{1,2}} & on\ I_2 \setminus I_{1,2} \end{cases}, \tag{18}$$

with the sets I_1, I_2 defined in (12) and $I_{1,2}$ in (13).

Algorithm implementing local RNN on I_{n1} and I_{n2},

1: repeat

2: $k := k + 1$

3: Call Loc-RNN

4: Exchange x_i^k between adjacent subdomains and compute $O_{1,2}$

5: until $\| \mathbf{g}_i^k - \mathbf{g}_i^{k-1} \| < eps$

3. Discussion

The aim of this work is only to illustrate the framework underlying such innovative ideas. Many aspects have been left intentionally unexplained. For example, regarding the size of the overlapping interval, we note that the majority of parallelization techniques first decompose the domain in nonoverlapping subdomains and then extend each subdomain with halo regions to enable stencil operations in a parallel context. Instead, the algorithm allows one to reduce communications among adjacent subdomains to only those nodes lying on the interfaces. In particular, if the widths of the halo regions are equal to one grid node wide, the halo nodes are actually the nodes of the overlapping region. Otherwise, halo regions contain more nodes than overlapping nodes and this means more communications among adjacent subdomains. Hence, the algorithm reduces communication times which depends on the number of inner nodes in overlap regions. In addition, the extra work performed on the overlapped region with an increasing size can be seen as the effect of a preconditioner on the overlapping region which can overestimate the solution [21].

Regarding the regularization parameter, we observe that the regularization parameter plays an important trade-off role. As is evident, when the regularization parameter overfitting occurs larger values are expected to produce reliable solutions even if the DD-step number increases. The choice of a good regularization parameter is one of the most important issues in solving unconstrained minimization problems and there exists a significant amount of research in the literature on the development of appropriate strategies for selecting regularization parameters. Parameter choice methods can be classified according to the input they require. There are two basic types: a-priori methods, requiring information about the noise level on data. The discrepancy principle, developed and analyzed by Morozov is the oldest one of them; data-driven methods, require no extra information. Generalized Cross-Validation (GCV), due to Wahba, is one of the most popular methods [22].

Citation: D'Amore L. A Model Decomposition-in-Time of Recurrent Neural Networks: A Feasibility Analysis. Trends Comput Sci Inf Technol. 2025;10(1):007-010. Available from: https://dx.doi.org/10.17352/tcsit.000091

Another issue is the employment of a dynamic load balancing scheme based on adaptive and dynamic redefining of initial decomposition. Specifically, in order to optimally choose the domain decomposition configuration, the partitioning into subdomains must satisfy certain conditions. First, the computational load assigned to subdomains must be equally distributed. Good quality partitioning also requires the volume of communication during calculation to be kept at its minimum. In [15] the authors employed a dynamic load balancing scheme based on adaptive and dynamic redefining of initial decomposition, aimed to balance load between processors according to data location. In particular, the authors focused on the introduction of a dynamic redefining of initial DD in order to deal with problems where the observations are non uniformly distributed and generally sparse.

The most significant finding is that we can solve several smaller problems improving the accuracy-per-parameter metric. Most importantly, subproblems can be solved in parallel, leading to a scalable algorithm where the workers locally exchange parameter updates via a nearest-neighbourhood communication scheme, which does not require a fully connected network. In contrast to other decomposition-in-time approaches, in our approach local solvers run concurrently from the beginning. Overall, the employment of the framework on hybrid high-performance computing systems seems to be a fruitful research area [23].

References

1. Schmidt RM. Recurrent neural networks (rnns): A gentle introduction and overview. arXiv:1912.05911. 2019. Available from: https://doi.org/10.48550/arXiv.1912.05911
2. Bishop CM. Pattern Recognition and Machine Learning. Springer; 2006. ISBN-10: 0-387-31073-8. Available from: https://link.springer.com/book/9780387310732
3. Dennis JE Jr, Schnabel RB. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. SIAM; 1996. Available from: https://books.google.co.in/books/about/Numerical_Methods_for_Unconstrained_Opti.html?id=RtxcWd0eBD0C&redir_esc=y
4. Bertsekas DP. Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey. arXiv:1507.01030v2. 2017. Available from: https://doi.org/10.48550/arXiv.1507.01030
5. Paine T, Jin H, Yang J, Lin Z, Huang T. GPU asynchronous stochastic gradient descent to speed up neural network training. arXiv:1312.6186. 2013. Available from: https://doi.org/10.48550/arXiv.1312.6186
6. Jager S, Zorn HP, Igel S, Zirpins C. Parallelized training of deep NN: Comparison of current concepts and frameworks. In: Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning. 2018;15-20. Available from: https://dl.acm.org/doi/10.1145/3286490.3286561
7. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521:436-444. Available from: https://www.nature.com/articles/nature14539
8. Lions JL, Maday Y, Turinici G. A parareal in time discretization of PDE's. C R Acad Sci Paris Ser I Math. 2001;332:661-668. Available from: https://www.scirp.org/reference/referencespapers?referenceid=2232887
9. Mayer R, Jacobsen HA. Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques and Tools. arXiv:1903.11314v1 [cs.DC]. 2019. Available from: https://doi.org/10.1145/3363554
10. Messi Nguelle T, Nzekon Nzeko'o AJ, Onana DD. Parallelization of Recurrent Neural Network training algorithm with implicit aggregation on multi-core architectures. 2024. hal-04542984v2. Available from: https://inria.hal.science/hal-04542984v2/document
11. Huang Z, Zweig G, Levit M, Dumoulin B, Oguz B, Chang S. Accelerating recurrent neural network training via two stage classes and parallelization. In: 2013 IEEE Workshop on Automatic Speech Recognition and Understanding. IEEE. 2013;326-331. Available from: https://ieeexplore.ieee.org/abstract/document/6707751
12. Nievergelt J. Parallel methods for integrating ordinary differential equations. Commun ACM. 1964;7:731-733. Available from: https://doi.org/10.1145/355588.365137
13. Parallel in time [Internet]. Available from: https://parallel-in-time.org/
14. Arcucci R, D'Amore L, Carracciuolo L. On the problem-decomposition of scalable 4D-Var Data Assimilation models. In: Proceedings of the International Conference on High Performance Computing and Simulation. Amsterdam, The Netherlands. 2015;5895942. Available from: https://ieeexplore.ieee.org/document/7237097
15. D'Amore L, Cacciapuoti R. Parallel Dynamic Domain Decomposition in Space-Time for Data Assimilation problems. J Phys Conf Ser. 2021;1828(1):012131. Available from: https://iopscience.iop.org/article/10.1088/1742-6596/1828/1/012131
16. D'Amore L, Cacciapuoti R. Space-Time Decomposition of Kalman Filter. Numer Math Theor Meth Appl. 2023;16(4):847-882. Available from: https://global-sci.com/article/90230/space-time-decomposition-of-kalman-filter
17. Elman JL. Finding structure in time. Cogn Sci. 1990;14(2):179-211. Available from: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog1402_1
18. D'Amore L, Murli A. Regularization of a Fourier series method for the Laplace transform inversion with real data. Inverse Problems. 2002;18(4):1185-1205. Available from: https://iopscience.iop.org/article/10.1088/0266-5611/18/4/315/pdf
19. Murli A, Cuomo S, D'Amore L, Galletti A. Numerical regularization of a real inversion formula based on the Laplace transform's eigenfunction expansion of the inverse function. Inverse Problems. 2007;23(2):713-731. Available from: https://iopscience.iop.org/article/10.1088/0266-5611/23/2/015/pdf
20. Gander MJ. Schwarz methods over the course of time. ETNA. 2008;31:228-255. Available from: https://etna.math.kent.edu/vol.31.2008/pp228-255.dir/pp228-255.pdf
21. Gander W. Least squares with a quadratic constraint. Numer Math. 1980;36:291-307. Available from: https://link.springer.com/article/10.1007/BF01396656
22. Golub GH, Heath M, Wahba G. Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter. Technometrics. 1979;21(2):215-223. Available from: https://doi.org/10.2307/1268518
23. Carracciuolo L, D'Amore L, D'Amore L, Murli A. Towards a parallel component for imaging in PETSc programming environment: A case study in 3-D echocardiography. Parallel Comput. 2006;32(1):67-83. Available from: https://doi.org/10.1016/j.parco.2005.09.001

Citation: D'Amore L. A Model Decomposition-in-Time of Recurrent Neural Networks: A Feasibility Analysis. Trends Comput Sci Inf Technol. 2025;10(2):007-010. Available from: https://dx.doi.org/10.17352/tcsit.000091