# Enhanced Distributed Computation for Machine Learning: Coded Strategies and Multidisciplinary Impact

*Xiaolin* Tian*

University of California, San Diego, La Jolla 92093, United States

**Abstract.** This article explores methods to accelerate distributed computation, focusing on its application in machine learning. It discusses two primary concepts: coded multiplication and data shuffling, along with a non-linear core to Random Access Memory (RAM) approach, presenting new avenues for future research. The challenges and future developments of coded computation in machine learning are examined, along with a general discussion on its applications across various scientific fields. The need for systems capable of handling massive data volumes has led to the adoption of large-scale distributed systems. These systems, however, introduce complexities such as straggler nodes and system failures, which can disrupt computational efficiency. Coded computation offers a solution by introducing redundancy that guards against system failures and enhances data noise mitigation. This paper highlights the strategic importance of coded matrix computation and data shuffling in improving fault tolerance and reducing communication costs within machine learning operations, thereby enhancing the overall efficiency and reliability of large-scale data analytics.

## 1 Introduction

In recent years, the exponential growth of data available for machine learning has necessitated the adaptation of computational paradigms to manage large-scale data analytics. This surge in data volume redefines not only everyday life and corporate operations but also expands the frontiers of scientific research, particularly in fields that have traditionally been constrained by the lack or difficulty of large-scale data collection [1]. This influx of data presents new challenges in managing and processing information efficiently, demanding advancements in distributed computing systems that can handle the increased scale and complexity.

Current distributed machine learning systems typically encompass three computational phases: storage, communication, and computation [2]. Despite their capabilities, these systems are plagued by inefficiencies such as straggler nodes, system failures, and communication bottlenecks, which can severely impede performance [3]. To address these issues, researchers have turned to innovative solutions like coded computation and data

*Corresponding author: xit058@ucsd.edu

shuffling. These methods introduce redundancy and optimize data handling to safeguard against failures and enhance overall system robustness. Despite the advancements, the unpredictability and unreliability of node performance within large systems continue to pose significant challenges, necessitating further exploration into more resilient and efficient computational strategies [4].

This paper delves into coded computation as a strategic approach to improve fault tolerance and reduce communication costs in machine learning operations. Specifically, it explores coded matrix computation and coded data shuffling as methods to enhance distributed system reliability and efficiency. The use of erasure codes and replication-based approaches offers potential solutions to mitigate the effects of stragglers and optimize system response times [5]. Additionally, this paper introduces a novel nonlinear core to Random Access Memory (RAM) approach that leverages the faster core to RAM communication to speed up processing times, which is particularly beneficial in multicore setups [6]. Through these investigations, the paper aims to provide a comprehensive analysis of current methodologies and project the trajectory of future developments in coded computation for machine learning applications across various disciplines.

## 2 Coded matrix computation

Coded matrix computation is a very commonly used strategy to improve the computational ability of an algorithm and increase the fault tolerance of the system. More specifically, in matic multiplication, codes are used to alleviate the influences of straggler, which are specific nodes that are slower at individual level than other nodes. What it does is to firstly divide raw data into multiple segments, and then distribute for storage and later computed at different nodes. Compared to having the machine process the raw information as a whole, it enables the computational system to perform parallel processing between bodies, which prevents the few stragglers slowing down the whole process [4].

In a large distributed computing environments, there are in general 4 steps of for coded matrix computation:

Data encoding: The first step is usually to encode the input data, then distribute the data as tasks to different nodes, for them to run the distributed processing parallelly. The encoding process can usually be achieved by matrix computation (or in some cases adding parity checks), or other mathematical operations.

Data calculation: The calculation at node level then happens in parallel. The nodes can run independently or interactively with each other.

Decoding; After the calculation, the machine decodes the result and tries to restore data back to how they were supposed to be.

Result generation: Finally they try to put up the final result and one set of coded computations is done.

## 3 Example of coded computation

There is an input data x, and machine M. After encoding, x is divided into $x_1$ and $x_2$. There are three workers, $w_1$, $w_2$, and $w_3$. $w_1$ is then assigned to decode $x_1$ and $x_2$ is assigned to decode $x_3$ and $x_2$ is assigned to to decode $x_1$ and $x_2$. In this machine, the original input is divided into 3 sub tasks. The outcome requires at least 2 workers to successfully decode the data, and restore x.

Straggler example: Let's take the previous example to consider the situation of straggler. At this time, for example, worker 2 runs into a delay or even total machine failure, and we have w1 and w3 working perfectly fine. Then to get the final outcome instead of waiting for

w2 to respond or being repaired, we can get $x_2$ by adding $x_1$ and $x_1$+$x_2$, which are the results from $w_1$ and w3. Then with $x_1$ and $x_2$ known, we can restore the information of x. Since $w_1$ and $w_3$ work in a parallel manner, the machine can automatically cancel the tasks assigned to $w_2$, and get ready for the next round of tasks, and in this set up, we can get the information from any of the first 2 nodes that finish computing. We can add more workers in different sets to increase the ability of failure tolerance.

Coded computation is a very broad idea, there are different approaches to get the task done, and the stragglers can also be caused by different reasons. For example, according to the author of, there can be unpredictable latency because of network, shared resource, maintenance activities, and even power limits, moreover, just like error can't be 100% eliminated from statistical computation. One approach that can deal with the straggler problem is to embed a straggler detection into the system. In this case, an algorithm can be used to detect the nodes running slow with its currently assigned work, and then reassign the work to another node that is more smoothly running the task, and avoiding waiting time of the straggler node. This approach can efficiently avoid having the straggler to work on the assigned task, but it still breaks the parallel execution continuity. Another approach that recently advanced to deal with the straggler problem is called replication-based approaches. What it does is by first replicating tasks and sending each node different replicas, then collecting outputs from the fastest nodes. This approach compared to the straggler detector algorithm guarantees the parallel continuity, therefore further reduces latency of stragglers. One feature of the replication-based approaches is that they potentially canceled the slower replicas from running, saving the computational capacity.

The use of erasure code to mitigate the effect of straggler is another recent topic that should acquire attention [5]. The idea is to utilize the nature that an erased code in many cases has a better chance to be recovered than a pure error. The application here is to treat an straggler as an erasure code, but instead of code here, its is an "erased node" and the machine then can tolerant erasure at some level, and start decoding with its partial result, without the need to wait for the "erased node'. Therefore mitigate the latency. More specifically, a simple MDA code, and some coded caching technique is used to speed up coded matrix multiplication and data shuffling.

In recent studies, there are new ideas of nonlinear approaches in speeding machine learning [6]. According to the author of the work, many studies (including those discussed above) have limited their focus on distributed setup, and omitted the fact that each node has more than just one core. In fact, there are cloud infrastructures that provide CPUs (Central processing units) up to 128 core, or GPUs (Graphics processing units) with 1000s of CUDA(Compute Unified Device Architecture) cores. So the basic idea is to gain speed in coded computation using the nature that a core to RAM communication is considerably faster than normal node to node computation. According to the author, after simulations of runtime performance of the computation schemes, there is a noticeable gain in speed for both coded and uncoded cases, which proves that the idea of running core to RAM computation can speed up the whole process.

Data shuffling. Data shuffling is one of the core elements for machine learning applications. It is a great approach when improving the statistical performance for learning algorithms. Coded data shuffling, compared to uncoded ways, provides advantages at reducing communication cost. The way it's done is by better utilizing cached data and reducing the storage required for data shuffling.

## 4 The Process of Data Shuffling in Distributed Machine Learning Systems

Data shuffling usually happens to randomly permute the partition of the data input into smaller batches, and then assign the batch to different nodes. There are typically 4 steps for it to be done.

Data Partitioning: This process is done at the master node after the input, and then the input is randomly permuted and the master node partitions it into several batches.

Batch Distribution: The masters then send each batch to the workers separately, and the respective workers then process different data sets to train the model.

Local training: This is where each individual node computes the allocated batches and trains themselves locally.

Model Aggregation: Each local model is then sent back to the master nodes, where they are averaged and used to generate the global model.

## 5 Example

Consider a dataset divided into four groups, which are assigned to four workers for processing. These data batches are randomly distributed among the workers, and after training, the process can be repeated with a different batch assignment. This approach allows for more efficient use of data but comes with significant drawbacks, primarily due to increased communication overhead. The repeated transmission of data batches contributes to higher latency and bandwidth consumption, which becomes more pronounced as the size of the dataset and the number of workers increase. Additionally, this process can be further delayed by straggler nodes, highlighting the need for more efficient strategies to minimize such disruptions.

Coded shuffling emerges as a practical solution to address the communication costs associated with data transmission. For example, consider a data matrix that is partitioned into several submatrices, distributed among workers. In traditional setups, after local training, the master node retransmits data to workers, which leads to redundant communication. With coded shuffling, workers utilize cached data and coded transmissions to reconstruct the required datasets without the need for direct retransmission. This approach effectively reduces communication overhead by leveraging the combination of cached and coded data, thereby facilitating the shuffling process.

An essential principle in data shuffling is ensuring that each processing unit handles different data in subsequent iterations to maximize learning efficiency. Many studies discuss the algorithmic "anatomy" of such processes, where nodes are assigned subsets of data, train models locally, and then repeat the process in iterative loops. This strategy provides fresh samples from the dataset, allowing nodes to achieve better statistical performance in the final outcome [7]. By utilizing one dataset to generate multiple unique samples for processing, the statistical power of the model is enhanced without the need for additional data.

The researchers in [8] proposed a novel approach to reduce communication costs using coding techniques. The coded shuffling algorithm is built on the coded caching scheme, which was originally developed to minimize communication rates in content delivery networks. Unlike traditional strategies, this method shuffles data points among computing nodes to improve statistical efficiency while encoding the data over real-valued representations (e.g., doubles or floats) rather than bits, as is typical in conventional methods. This distinction enhances the effectiveness of the approach by simultaneously optimizing communication and computational performance.

## 6 For future studies

Eventually, there are a few untested ideas for future studies. All these ideas include topics that have been discussed in the paper and other concepts that we haven't touched based on in this article.

Alternative coding techniques: Reed-Solomon codes and Turbo codes are not discussed in this article, but they may offer different trade-offs in terms of redundancy and performance when combining coded computation techniques. There may be different outcomes when dealing with system noises in a distributed system.

Convergence Analysis: Convergence properties of machine learning algorithms may play some interesting role in coded data shuffling. There are existing works that studies convergence rate, for example stochastic gradient descent (SGD) under different communication models, and combining them with data shuffling techniques will be an interesting topic.

Adaptive Coded Computation: There are adaptive coding strategies that have algorithms that have the ability to check on the worker nodes in real-time and evaluate the worker performance in real-time. Then the machine can decide when to use coded computation and when to not use it based on the system noises that it detects, and better reduce latency.

Hybrid approaches. When at low-bandwidth scenarios, a hybrid model may contain coded computation and strategies such as gradient compression, or quantization method.

Cross-Layer Optimization: Optimization process done across the three layers (computation, communication, and storage) using a unified coding framework. An algorithm that can measure impacts of coding simultaneously across three layers can be developed.

## 7 Social science application

The abundance of data available for social science research is now greater than ever, which allows researchers to mine and abstract in large and small scales. Compared to traditional, less quantitative and more deductive approaches of social science throughout its history, all this new resource gives social science a more sequential, and inductive approach for people to better interact with data.

Empirical studies in social sciences, due to the hardness of data collection, and scarce usage of computational power, have made researchers develop statistical methods that levaging small amounts of data and even less computing power. The abundance of data now offers more flexible algorithms and statistically powerful tools that build new machine learning approaches for social scientists and they usually include tasks like discovery, measurement, causal inferences, and prediction [9].

A deductive approach in common practice of social science is that research should be based on evidence that is already given, that is before viewing or collecting data, the researchers should have a clear idea of the theories that they will derive, and design the variables, with an appropriate measure of the concepts, then develop a set of hypotheses and run the experiment.

Social scientists are now discussing agnostic approaches while using machine learning to make inferences. The general idea is that unlike in computer science applications, that there can be one particular best model for prediction, there are many instances that for one particular study and its steps, there is not a correct or true model that should be used, either there is one single best model [10]. This approach is efficient when there is less data, and it actually brings a lot of advantages including reducing false discoveries, and it is powerful especially when a theory and its implications is sound [11]. However, there are still new discoveries and questions arising from the data collected from deductive reasoning that can’t be well explained based on current existing knowledge, therefore an inducting approach

heavily involving qualitative analyses is very necessary [12]. Machine Learning then in this case is a great tool for refining and developing new theories based on its strong power of discovery prediction, causal inference, and measurement.

An agnostic view that is used in is sharply different compared to structural approach with using machine learning in method. A structural approach posits the existence of an underlying true data generating process, and if there is assumption made about underlying true model of data, the model selecting process would be easier. However, this process often involves unrealistic assumptions about data generating. In a intertwined study that involves a large social network, it is hard to make the most accurate predictions, then machine learning methods then can be applied to discover and lead to new research question, hypotheses and insights, and in this discovery process, there can be many methods used, until the research question is positive, there can be models used in measures, and evaluation of witch model is more preferable can be done.

## 8 In soft materials engineering

Machine learning and data science is now also routinely used in material science nowadays to replace conventional methods of analyzing data in large size and dimensionality that fails. Machine learning and data science in material science plays an important role in identifying and extract rend Ran patterns within voluminous data sets and can also perform traversals of high-dimensional phase spaces [13].

There are 5 popular approaches that machine learning is employed in soft and biological material engineering(sm): Principal component analysis (PCA), Independent component analysis, diffusion maps, SVM(support vector machines), and relative entropy.

PCA is the abbreviation for principal component analysis, along with many other names including KLTKarhunen- Loeve transform, proper orthogonal decomposition (POD), and factor analysis [14-17]. It is an unsupervised linear dimensionality reduction technique that identifies within a high dimensional input data set a low-dimensional hyperplane containing the preponderance of variance in the data. Independent component analysis (ICA) is also an unsupervised linear dimensionality reduction technique, the difference between ICA and PCA is that ICA decomposes the high dimensional data set into a statistically independent component [18]. Diffusion maps(dMaps) compared toICA and PCA is a nonlinear technique that belongs to the class of manifold learning methods. It can form low dimensional projections onto curvilinear manifolds, and made dMaps a powerful tool. SVM is called support vector machines, and it is a supervised linear binary classification technique that partitions data sets into two disjoint classes and then learns the deterministic rules and relative entropy (relEnt) coarse-graining is an information theoretic measure of similarity between two probability distributions [19].

## 9 In environmental research

Machine learning and data science is also increasingly used in environmental research when processing large data sets, and they need to identify causal correlation and causality. The new computational approaches offer a strong tool to do all the tasks. However, this new approach may also bring problems if we couldn't select the best strategies to deal with.

In the article, there are two reference paradigms that describe the best practice of machine learning models in the environment [20]. The first is called "recommended minimum requirements" and the other one is called "recommended best practice". It would be essential to discuss the rationale of the variables or feathers and check if there is a feature X that may

cause data leakage. In other words the objective for the best practices paradigm is to establish a foundational step for building a supervised machine learning model with essential data management strategy. One of the important steps is to ensure proper feature selection, therefore to avoid data leakage one example can be time-series, which may need special preprocessing methods to deal with.

Another recommendation is using validation data for hyperparameter optimization with a test data reaction of 0.1-0.4, noting that preprocesses only the pre-training set before applying methods on the calideation set. There should be at least two evaluation metrics and comparisons with statistical models are advised to assess model performance and explainability and causal relationships.

The second paradigm is called the "Best practices paradigm" and it talks about more strategies and rigorous criteria that could be used in addition to the minimum requirement talked above.

One thing that is essential for excluding low quality data and abnormal data is to apply quality assurance / quality control (QA/QC). There also should be steps being included in managing data quality by filtering outliers, establishing alternative analysis endpoints, and adjusting sample size when it's necessary. Cross violation or train validation techniques can be used with grid search of metaheuristic methods. Another strategy is to apply weighted metrics for imbalance data, comparison across statiscally and machine models. There should also be additional steps in examining causality to increase model reliability.

## 10 Conclusion

This paper has thoroughly explored the integration of coded computation strategies within distributed machine learning systems, focusing primarily on coded matrix computation and data shuffling techniques. The adoption of these methodologies significantly enhances system robustness by improving fault tolerance and reducing communication overhead, which are critical in managing the complexities of large-scale data analytics. Our exploration underscores the pivotal role of redundancy in coded computation to guard against potential system failures and straggler impacts that commonly afflict distributed systems. Looking forward, the paper identifies several promising avenues for future research. The application of advanced coding techniques such as Reed-Solomon and Turbo codes offers potential for further improvements in redundancy and system performance. Additionally, the integration of adaptive coded computation strategies that dynamically adjust based on real-time system performance could lead to more efficient data processing frameworks. Another exciting prospect is the development of hybrid approaches that combine coded computation with other optimization strategies like gradient compression, particularly useful in low-bandwidth scenarios. Moreover, cross-layer optimization presents a novel research trajectory, proposing a unified coding framework across the computation, communication, and storage layers of distributed systems. This approach aims to harmonize the impact of coding across these layers, potentially leading to ground breaking enhancements in processing speed and efficiency.

In conclusion, while the current strategies enhance the efficiency and reliability of machine learning operations across various scientific disciplines, there remains a vast landscape of unexplored potentials in coded computation. Future studies will undoubtedly build on the foundational concepts discussed here, propelling the evolution of machine learning technologies to new heights.

## References

1. K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, K. Ramchandran, Speeding up distributed machine learning using codes. IEEE Trans. Inf. Theory **64,** 1514–1529 (2017)
2. J. Dean, L.A. Barroso, The tail at scale. Commun. ACM **56**, 74–80 (2013)
3. M. Zaharia, A. Konwinski, A.D. Joseph, R.H. Katz, I. Stoica, Improving MapReduce performance in heterogeneous environments. Proc. 8th USENIX Symp. Operating Syst. Design Implement. (OSDI), 29–42 (2008)
4. X. Zhu, Z. Zhao, X. Wei, X. Wang, J. Zuo, Action recognition method based on wavelet transform and neural network in wireless network. Proc. 2021 5th Int. Conf. Digital Signal Processing, 60-65 (2021)
5. X. Zhu, C. Guo, H. Feng, Y. Huang, Y. Feng, X. Wang, R. Wang, A Review of Key Technologies for Emotion Analysis Using Multimodal Information. Cognitive Computation, 1-27 (2024)
6. Y. Zhang, H. Xu, X. Zhu, Z. Zhao, J. Zuo, Detection and Quantization Technique of Optical Distributed Acoustic Coupling Based on φ-OTDR. J. Shanghai Jiaotong Univ. (Science) **25(2),** 208-213 (2020)
7. B. Recht, C. Ré, Parallel stochastic gradient algorithms for large-scale matrix completion. Math. Program. Comput. 5, 201–226 (2013).
8. M.A. Maddah-Ali, U. Niesen, Fundamental limits of caching. IEEE Trans. Inf. Theory **60**, 2856–2867 (2014)
9. J. Grimmer, M. Roberts, B. Stewart, Machine learning for social science: An agnostic approach. Annu. Rev. Polit. Sci. **24**, 395–419 (2021)
10. Y. Huang, R. Wang, M. Ju, X. Zhu, Y. Xie, Reconstructing global daily XCO2 at 1°×1° spatial resolution from 2016 to 2019 with multisource satellite observation data. J. Appl. Remote Sensing **18(2)**, 028502-028502 (2024)
11. M. Humphreys, R. Sanchez de la Sierra, P. Van der Windt, Fishing, commitment, and communication: A proposal for comprehensive nonbinding research registration. Polit. Anal. **21**, 1–20 (2013)
12. Z. Zhao, X. Zhu, X. Wei, X. Wang, J. Zuo, Application of Workflow Technology in the Integrated Management Platform of Smart Park. Proc. 2021 IEEE 4th Adv. Inf. Management, Communicates, Electronic and Automation Control Conf. (IMCEC), vol. **4**, 1433-1437 (2021)
13. X. Wei, X. Zhu, X. Wang, Z. Zhao, J. Zuo, Fuzzy Fault Tree Analysis Method and Its Application in Fault Diagnosis of Denitration System in Thermal Power Plant. Proc. 2020 8th Int. Conf. Information Technology: IoT and Smart City, 227-232 (2020)
14. X. Zhu, Y. Zhang, Z. Zhao, J. Zuo, Radio frequency sensing based environmental monitoring technology. Fourth Int. Workshop on Pattern Recognition, vol. **11198**, 187-191, SPIE (2019)
15. H. Xu, X. Zhu, Z. Zhao, X. Wei, X. Wang, J. Zuo, Research of Pipeline Leak Detection Technology and Application Prospect of Petrochemical Wharf. 2020 IEEE 9th Joint Int. Information Technology and Artificial Intelligence Conf. (ITAIC), vol. **9**, 263-271 (2020)
16. Z. Zhao, Y. Peng, X. Zhu, X. Wei, X. Wang, J. Zuo, Research On Prediction Of Electricity Consumption In Smart Parks Based On Multiple Linear Regression. 2020

IEEE 9th Joint Int. Information Technology and Artificial Intelligence Conf. (ITAIC), vol. 9, 812-816 (2020)

17. K. Pearson, On lines and planes of closest fit to systems of points in space. London Edinb. Dubl. Philos. Mag. J. Sci. **2**, 559–572 (1901)
18. A. Hyvärinen, J. Karhunen, E. Oja, Independent Component Analysis, Vol. **46**. Wiley, New York, 2004
19. R. Mullin, Partnership applies deep learning to very big data: GlaxoSmithKline teams with government labs on computational models for drug discovery. Chem. Eng. News **95**, 31–32 (2017)
20. J. Zhu, M. Yang, Z. Ren, Machine learning in environmental research: Common pitfalls and best practices. Environ. Sci. Technol. **57**, 46 (2023)