Received 29 April 2024; revised 3 June 2024; accepted 25 June 2024. Date of publication 4 July 2024; date of current version 25 July 2024.

Digital Object Identifier 10.1109/OJCOMS.2024.3423362

Coded Federated Learning for Communication-Efficient Edge Computing: A Survey

YIQIAN ZHANG [1,2,3], TIANLI GAO [1,2,3], CONGDUAN LI [1,2,3] (Senior Member, IEEE), AND CHEE WEI TAN [4] (Senior Member, IEEE)

[1]School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen 518107, China

[2]State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China

[3]Shenzhen Key Laboratory of Navigation and Communication Integration, Sun Yat-sen University, Shenzhen 518107, China

[4]School of Computer Science and Engineering, Nanyang Technological University, Singapore 639815

CORRESPONDING AUTHOR: C. LI (e-mail: licongd@mail.sysu.edu.cn)

This work was supported in part by the National Science Foundation of China (NSFC) under Grant 62271514; in part by the Science, Technology and Innovation Commission of Shenzhen Municipality under Grant JCYJ20210324120002007 and Grant ZDSYS20210623091807023; in part by the Research Fund of State Key Laboratory of Public Big Data, Guizhou University under Grant PBD2023-01; and in part by the Ministry of Education, Singapore, under its Academic Research Fund Grant AcRF RG91/22 and NTU Startup

(Yiqian Zhang and Tianli Gao contributed equally to this work.)

ABSTRACT In the era of artificial intelligence and big data, the demand for data processing has surged, leading to larger datasets and computation capability. Distributed machine learning (DML) has been introduced to address this challenge by distributing tasks among multiple workers, reducing the resources required for each worker. However, in distributed systems, the presence of slow machines, commonly known as stragglers, or failed links can lead to prolonged runtimes and diminished performance. This survey explores the application of coding techniques in DML and coded edge computing in the distributed system to enhance system speed, robustness, privacy, and more. Notably, the study delves into coding in Federated Learning (FL), a specialized distributed learning system. Coding involves introducing redundancy into the system and identifying multicast opportunities. There exists a tradeoff between computation and communication costs. The survey establishes that coding is a promising approach for building robust and secure distributed systems with low latency.

INDEX TERMS Coding, distributed machine learning, federated learning, distributed computing, edge computing.

I. INTRODUCTION

DISTRIBUTED machine learning (DML) enables the processing of vast datasets and the training of complex models across distributed environments in today's data-driven world. By harnessing the collective computational power of networked devices, DML empowers organizations to tackle challenges in areas ranging from natural language processing (NLP) and computer vision to healthcare and finance. Yet, with the continual expansion in the scale and complexity of distributed systems, conventional approaches to distributed computing have become progressively inefficient and costly. In particular, conventional approaches to distributed computing struggle to cope with the challenges posed by slow worker nodes and links, often referred to as "stragglers," because the server node must await the completion of all worker tasks, thereby impeding the overall computational speed, as shown in Figure 1. Coded DML, emerging as a promising paradigm, seeks to revolutionize traditional distributed computing methods by incorporating coding techniques such as repetition-based techniques or erasure codes to mitigate the impact of stragglers and optimize resource utilization. By leveraging redundancy and encoding strategies, coded DML offers a transformative framework for faster convergence, improved fault tolerance, and enhanced scalability in distributed learning systems.

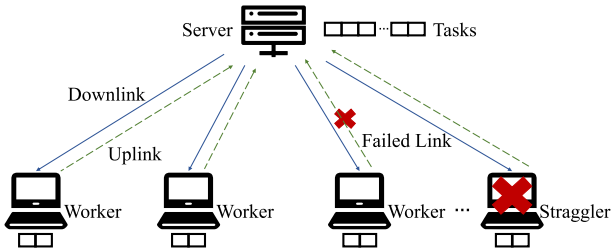


FIGURE 1. Sketch of Distributed Machine Learning.

One notable strategy in DML is coded computing, which revolutionizes data processing by distributing data across multiple nodes within the system. This methodology optimizes resource utilization by enabling each node to handle only a fraction of the data, thus enhancing efficiency without compromising accuracy. Through mathematical techniques like erasure coding and network coding, coded computing offers fault tolerance and minimizes communication overhead among nodes. Widely adopted across various machine learning domains such as image classification, speech recognition, and NLP, coded computing consistently delivers reduced computation and communication costs while upholding superior accuracy and resilience. As DML continues its growth trajectory, the integration of coded computing stands poised for the scalability and effectiveness of these systems.

In a dynamic edge computing landscape characterized by heterogeneous computational capacities, storage capabilities, and communication resources among edge devices, [1] introduces coded edge computing (CEC). Unlike conventional approaches where results from slow workers are taken as erasures, CEC leverages any coded computation results returned within the deadline. By integrating data encoding with computation decoding techniques such as quantization and sphere/multi-stage decoding via modulo operations, CEC enables the utilization of partially completed computations. This innovative approach mitigates communication bottlenecks, enhances system robustness against failures, and reduces run times.

Training machine learning (ML) models typically demand abundant labeled data, yet many fields grapple with limited or unlabeled data. Take, for instance, a product company that possesses information on clients' purchases but lacks insights into their purchasing power, which is crucial for training recommendation algorithms. This scarcity of data poses a significant challenge to ML development. Conventional DML systems require clients to upload original raw data to a central server for training tasks, which occupies considerable communication bandwidth, resulting in prolonged transmission delays. Uploaded data may contain sensitive client information, such as facial images, personal assets, or medical records, thus exacerbating privacy risks. Heightened awareness of privacy issues has prompted legislative efforts to regulate the handling of private data, as exemplified

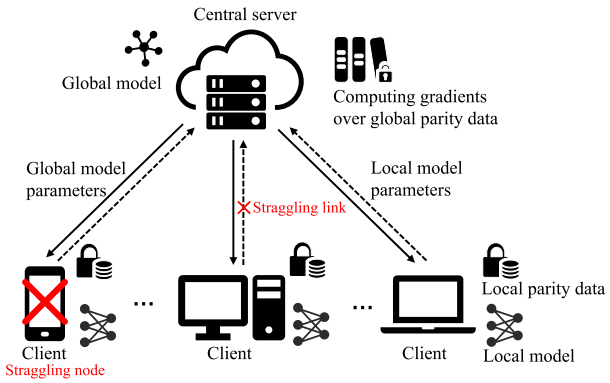


FIGURE 2. Illustration of the Coded Federated Learning (CFL) [5] scheme in FL system. Each client generates parity data to transmit to the central server before training for compensating the straggling nodes and links. The central server computes gradients over global parity data, which is combined with the received gradients to obtain an overall global model.

by Europe's General Data Protection Regulation (GDPR) enacted in 2018 [2] and China's Cybersecurity Law of 2021 [3]. These regulations, while essential for safeguarding privacy, present additional hurdles for ML training.

To address the challenges of training ML models on decentralized data, Federated Learning (FL) [4] has been proposed. FL is a DML setting where a cluster of edge devices, orchestrated by a central server, jointly train a machine learning model while maintaining data decentralization and locality. Rather than transmitting data to the central server, FL facilitates training directly on the edge devices where data originates. The model updates, such as weight adjustments or gradients, are then transmitted to the central server for aggregation. The central server consolidates all received data and produces an updated model, which is subsequently transmitted back to the clients for additional training. This iterative process persists until the global model attains the desired performance level.

FL frequently encounters communication bottlenecks due to the participation of numerous clients, such as smartphones and vehicles, which communicate extensively with edge devices and central servers. Many FL training scenarios involve mobile devices with limited computational capabilities, like wearables and autonomous vehicles, often leading to node failures during training. Consequently, novel methods and techniques are developed to enhance the communication efficiency of FL, with coding emerging as a promising solution. For instance, error correction codes (ECCs) can bolster the robustness of FL systems against node or link failures by introducing redundant data alongside the original data [5]. When computing nodes or communication links fail during training, data loss or incomplete computations may occur, adversely affecting model accuracy. To address this challenge, [5] proposes a novel approach called Coded Federated Learning (CFL), where clients locally generate redundant parity data, subsequently transmitted to the central server to mitigate the straggling problem, as depicted in Figure 2. By leveraging ECC, FL systems can better manage

TABLE 1. A summary of coding techniques used for distributed machine learning in the paper.

Learning	Scenarios	Content	Reference
Distributed Coded Machine Learning	Coding in DML	Gradient Coding	Section II-A
		Encoded Distributed Optimization	Section II-B
		Other Coding Scheme	Section II-C
	Coded Edge Computing in DML	Diversity Gain	Section III-A
		Learning-based Coded Computation	Section III-B
		Cloud ML System	Section III-C
		Privacy and Security in DML	Section III-D
	Coding in FL	Straggling	Section IV-A
		Communication Loads	Section IV-B
		Privacy and Security in FL	Section IV-C
	Evolution of Codes and Real-World Applications	Evolution of Codes	Section V-A
		Real-World Applications	Section V-B

TABLE 2. Key findings in Section II Coding in DML.

Coding Techniques	Content	References
Gradient Coding	Gradient coding: use MDS codes to ensure any k out of n machines can recover	[6] [7]
	Improved Gradient Coding Based Algorithms	[8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29]
Encoded Distributed Optimization	The data variables are redundantly encoded in a linear fashion and distributed among the nodes	[30] [31] [32] [33] [34] [35] [36]
Other Coding Schemes	Some other codes such as source coding, Slepian-Wolf coding, polar codes, etc.	[37] [38] [39] [40]

errors during data transmission or storage, thus minimizing the risk of inaccurate results. Additionally, coding can be applied to enhance the efficiency, reliability, and security of data transmission between communication endpoints. This process involves converting data into a specific format before transmission, ensuring successful interpretation and reconstruction at the receiving end.

This survey conducts a comprehensive literature review on current researches that integrate coding techniques in distributed machine learning and federated learning systems, highlighting coding as a promising technique in enhancing system performance, communication efficiency, and privacy. It gives not only a comparison but also real-world applications such as mobile edge computing or IoT devices. Table 1 provides a comprehensive overview of coded distributed machine learning as discussed in this paper. In Section II, we delve into research that applies coding techniques to expedite DML. Section III focuses on the investigation of coded edge computing in the context of DML. Moving forward, Section IV explores the realm of coding in FL. Section V summarizes the evolution of codes in DML and gives the real-world applications such as mobile edge computing or IoT devices along with discussions on potential avenues for future research. Section VI concludes the survey.

II. CODING IN DML

In DML systems, the issue of communication latency looms large, particularly in the context of slow-performing workers or straggling links. To address this challenge, numerous coding methods have been introduced to mitigate the impact of stragglers and optimize communication efficiency. Table 2 lists the key findings from the researched references regarding coding in ML.

A. SPEEDING UP DML WITH GRADIENT CODING

Coding has been used in distributed computing in various scenarios. In distributed learning scenarios, the worker nodes typically perform gradient computations on their respective local datasets, while the central server aggregates these local gradients to update the global model. Coding has been introduced into DML by Lee et al. in 2018 [6], where they tried to reduce latency and further speed up learning by tackling the problem of noise like straggler nodes, system failures, or communication bottlenecks with codes. Also, they propose coded shuffling for data shuffling, which can significantly reduce the price.

Maximum distance separation (MDS) codes are a class of error-correcting codes that are designed to provide the maximum possible distance between codewords. In the context of coding theory, "distance" refers to the number of

positions at which two code words differ. The higher the distance, the better the code is at correcting errors. Reed-Solomon (RS) codes [41] is one of the most well-known and widely used types of MDS codes. These codes are based on polynomial algebra and have efficient decoding algorithms.

The objective of [7] is to avoid stragglers when they are unknown by using *gradient coding* where every node sends a linear combination of the vectors, enabling recovery of the desired vector with any k out of n machines. This approach ensures fault tolerance for slow or failed machines, with negligible computation cost compared to communication overhead. Although [7] and [6] address the problem of worst devices (stragglers) to speed up the DML, they encounter challenges in handling straggler-related issues within the underlying calculations of distributed neural networks, especially concerning model parallelism. This is due to the nonlinear nature of neural networks, compounded by activation functions between layers, which conventional coded computation struggles to accommodate, particularly during the backward pass and gradient update processes. To solve these problems and reduce the communication load inherent in distributed systems, [8] proposes a coded computing scheme that integrates neural networks and model parallelism into underlying calculations. Such coded parallelism schemes have better performance in both computation and communication and offer better stability even with unstable devices. Additionally, in [9], coded stochastic incremental alternating direction method of multipliers (csI-ADMM) algorithms are used to tolerate link failures and stragglers by exploiting redundancy. Error-control coding is employed to enable any k out of n coded blocks to recover k message blocks. The two MDS-based coding schemes, fractional repetition scheme and cyclic repetition scheme are adopted as in [7].

In [10], a novel gradient method termed lazily aggregated gradient (LAG) is introduced for gradient-based algorithms in DML. LAG incorporates adaptive gradient skipping, effectively identifying slowly changing gradients and reusing outdated ones to economize on communication and computation costs. Building upon this innovation, [11] presents lazily aggregated gradient coding (LAGC), which combines Gradient Coding (GC) with LAG. By dividing all the workers into groups and applying coding within each group, LAGC regards the group as a worker in LAG. This hybrid approach offers a tradeoff between GC's robustness against stragglers and the efficiency of LAG. LAGC operates as GC when all workers are combined into one group, and functions as LAG when each worker represents a separate group.

Later research on gradient coding and approximate gradient coding has demonstrated how redundancy can be incorporated into distributed gradient descent to ensure convergence, even when some workers are slow or unresponsive. For instance, [12] proposes stochastic gradient coding (SGC). SGC requires a small amount of redundancy to handle numerous stragglers, and when the number of stragglers is large, it outperforms existing approximate gradient codes. It is a form of approximate gradient coding designed to facilitate distributed gradient descent-based machine learning algorithms across n workers, even when stragglers are present. In SGC, by incorporating minimal data redundancy, when the few fastest workers complete the assigned computations, masters can move to the next iteration.

Random linear codes are employed in [13] to optimally encode and balance the load of training data, avoiding an explicit decoding step. The heterogeneous coded gradient descent (HCGD) algorithm proposed in this research explicitly models heterogeneity and stochastic behavior of computing and communication resources. By leveraging statistical insights, HCGD strategically injects optimal redundancy into training data using random linear codes, simultaneously balancing the load during the encoding process. The framework reduces average epoch time, effectively mitigating the straggler effect. Numerical experiments using synthetic data demonstrate a convergence rate ten times faster than repetition coding, dependent on the underlying heterogeneity of the mobile edge computing (MEC) platform. The proposed methodology incurs a one-time cost for encoding and transmitting the training data to worker devices.

In the context of the distributed learning paradigm featuring a central parameter server (PS), as highlighted in [14], the PS holds complete access to the dataset. This setup utilizes coding techniques to distribute data partitions among distributed nodes. Leveraging the broadcast nature of wireless networks, this approach enables nodes in overlapping regions to concurrently transmit their results to multiple access points (APs). Specifically, devices transmit their gradients to APs, with those situated in overlapping AP areas broadcasting to all APs. Subsequently, APs encode the received gradients and relay them to the PS. This hierarchical architecture effectively mitigates straggler issues and yields significant performance enhancements compared to traditional gradient coding methods.

In [15], coded redundancy is employed in communication links, where each of the n_e clients transmits coded gradient updates to n_h helpers. To align corresponding gradient components from different clients, each client divides its gradient update into unique components and sends them to multiple helpers. This approach, known as aligned repetition coding (ARC), minimizes the communication loads between clients and helpers. The gradient of each client is partitioned and an MDS code is applied to the partitions. With an identical generator matrix across clients, each parity corresponds to a unique helper for all clients. Only when $(n_h - s)$ helpers receive messages from the same set of client nodes, partial aggregations of helpers are beneficial. Such aligned minimum distance separable coding (AMC) is to minimize helpers-to-master communication loads. The hierarchical aggregation process is designed to tackle up to s straggling links out of the n_h helper links per client, where s represents the resiliency threshold. Two hierarchical

aggregation methods are proposed, ensuring a robust and efficient network architecture.

In [16], the focus lies on tolerating and leveraging stragglers in heterogeneity, leading to the proposal of Heter-aware and DHeter-aware algorithms. The Heter-aware algorithm assigns data partitions to workers based on their computing capability adaptively. It relies on the outcomes from fast workers for decoding while disregarding the results from stragglers. In contrast, the DHeter-aware algorithm caches the results of stragglers instead of discarding them. Even as stragglers continue to compute gradients, the master node decodes the gradients from the cache once a sufficient number of coded gradients have been accumulated. Then the parameter is upgraded. Those gradients sent by stragglers will be cached and used for future decoding as long as their delayed rounds are less than a certain threshold.

A novel scheme known as multi-message communication (MMC) is introduced in [17], enabling workers to transmit multiple computations to the master during each iteration, unlike other schemes that transmit only one result. The general coded distributed gradient descent (DGD) they propose is to make the most of non-stragglers at the price of increasing computation load. Additionally, they tell how to balance computation time with communication load, offering insights into optimizing performance in distributed systems.

Gradient coding with multi-message communication (GC-MMC) is proposed in [18], where a static clustering technique is adopted to improve the performance of GC. It allows workers to transmit partial gradients even when the assigned computations are not completed, leveraging the computational capacity of stragglers. Workers are organized into disjoint clusters of equal size, leading to reduced computation time. There is a tradeoff between the computation load and completion time. One advantage of clustering is that, with uniformly distributed stragglers among clusters, the system can tolerate more stragglers with an increased number of clusters. Conversely, in cases of non-uniform straggler distribution, performance may suffer.

In GC and GC with static clustering, the designed codes are engineered to ensure perfect gradient reconstruction for every iteration, even in worst-case scenarios. To further improve GC, [19] introduces a new scheme, allowing for the dynamic selection of codes based on previous straggling behavior. This innovation, termed GC with dynamic clustering (GC-DC), addresses performance degradation issues observed in [18], particularly when stragglers are non-uniformly distributed. By dynamically forming clusters with uniformly distributed stragglers, and leveraging the time-correlated nature of straggling behavior, GC-DC effectively reduces completion time for both homogeneous and heterogeneous worker setups. Detailed dynamic codeword assignments are of two stages:

- Data assignment. It executes only once when the training begins and assigns mini-batches to workers within their memory constraints;
- Codeword assignment. It assigns codes to workers at the beginning of every iteration at time slot t by the PS according to the past stragglers' behaviors in the past $t-1$ time slots.

The focus of [20] is on non-persistent stragglers and a novel approach to leverage their computation results for expediting completion time is introduced. Their proposed Multi-Message Gradient Coding scheme does not rely on data encoding, instead, evaluations are encoded across multiple communication messages. This strategy offers increased flexibility in balancing communication load and completion time, making it applicable to broader problems. Moreover, its adaptability to evolving system requirements without necessitating re-encoding of the entire dataset enhances its practical utility.

A gradient coding framework called Communication-efficient Fractional Repetition Gradient Coding (CommFR-GC) is proposed in [21], to improve communication efficiency. The framework employs linear codes to construct encoding and decoding functions. Notably, the framework distinguishes between different performances based on the following three characters of the codes. This distinction enables the balancing of the straggler threshold and communication overhead, all while maintaining lower decoding complexity and greater numerical stability:

- block-length l of a code $\rightarrow$ computation load;
- dimension m $\rightarrow$ communication overhead;
- minimum distance s $\rightarrow$ straggler tolerance.

There are three phases in CommFR-GC, placement, encoding, and decoding. This framework proves that employing a random Gaussian matrix to create an MDS code results in an optimal gradient code with improved numerical stability.

In [22], a novel approach involving the encoding of the second moment of data using a low-density parity-check (LDPC) code is proposed for distributed gradient descent, specifically designed to counteract the impact of straggling processors, particularly suited for the squared-loss function. The iterative decoding algorithms proposed exhibit minimal computational overhead, extendable to various loss functions, and offer convergence guarantees under a random model of stragglers.

There are two features of gradient coding: i) to recover full gradients from partial workers; ii) to recover partial gradients from arbitrary workers. An illustration of the latter concept can be found in the anytime coding scheme proposed by [23] for distributed computation. A computational job is decomposed, and linear ECC is used for the assigned sub-tasks in multiple workers. Anytime coding allows computation tasks to be stopped at any time, combining the results from completed processors to derive an approximate solution. Unlike traditional MDS coding, where the master must await the return of results from sufficient workers to estimate the computation accurately, anytime coding provides an approximation computation based on the outcomes from currently completed workers. The longer the

time the master waits, the more completed processors there are, resulting in increasingly accurate results, highlighting the inherent accuracy/latency tradeoff. The focus of [24] is on the ii) and proposes ignore-straggler gradient coding (IS-GC) to recover as many gradients as possible from arbitrary gradients. It achieves larger gradients recovered compared to GC and ignore-straggler stochastic gradient descent (IS-SGD) [42]. This paper also applied hybrid repetition (HR), a hybrid scheme of fractional repetition (FR) and cyclic repetition (CR), which achieves the trade-off between the more gradients recoverable of FR and the flexible choices of parameters of CR.

In the pursuit of optimizing redundancies and leveraging the computational potential of non-persistent stragglers within gradient coding, [25] fully utilizes the incomplete results generated by these non-persistent stragglers, commonly referred to as partial stragglers. In large-scale machine learning tasks with L model parameters, a coordinate gradient coding scheme is introduced, where L coding parameters are assigned to each coordinate, potentially diversifying the redundancies across the model parameters. The optimization involves minimizing the expected overall runtime while maximizing the completion probability concerning the L coding parameters associated with coordinates.

Building on previous work by [6] and [23], [26] proposes a sequential approximation framework for distributed optimization problems. This approach solves a sequence of optimization problems that approach the solution of the original problem, designed to require less computation time than solving the original problem. The method is designed to address latency caused by "stragglers" in distributed computation systems and guarantees useful computation results even with uncertain latency.

A novel approach aimed at minimizing communication delays through the utilization of a sequential gradient coding scheme that codes across time and workers is proposed in [27]. The introduced product-matrix (PM) sequential gradient coding scheme uses the temporal dimension to handle erasure patterns that follow the (n, S, W)-isolated erasure model. This provides more flexibility than the traditional GC method where only cross-coding among workers is allowed, and better resilience against communication delays while maintaining the same computational load.

A two-stage coding scheme in DML is proposed by [28] to recover partial gradient, and thus enhance accuracy and resource utilization, considering both computation and communication phases. In the computing phase, a two-stage dynamic coding approach is introduced. In the first stage, a subset of workers computes partial gradients, while the remaining workers initiate computation in the second stage, based on the completion status of the first stage. Additionally, to optimize the balancing of admission data in terms of fairness and throughput during the communication phase, a perturbed Lyapunov function is designed.

Also, (n, q)-MDS codes are used in [29] to mitigate the impact of stragglers by allowing the master node to utilize partial computation outputs. Such schemes consistently outperform uncoded approaches in terms of runtime under both a constant-speed model and a two-state Markov chain model, and they exhibit robustness by eliminating the need for careful parameter selection of q when workers are at varied speeds.

B. ENCODED DISTRIBUTED OPTIMIZATION

Machine learning encompasses various problem domains, such as linear regression, support vector machines, compressed sensing, and maximum-likelihood estimation, all of which entail large-scale optimization challenges. Traditional distributed optimization methods distribute these problems across multiple nodes, but they can suffer from delays or node failures, resulting in a slowdown that negates the benefits of distributed methods. To address this issue, a new framework for encoded distributed optimization is proposed where the data variables in the optimization problem are redundantly encoded linearly and distributed among the nodes. Large-scale machine learning and data analytics problems often require distributed optimization, which can be vulnerable to network and node failures. To address this, [30] proposes a new approach using the linear encoding of data variables to add redundancy and increase robustness against failures. This method allows for close approximations to solutions even under node failures, with equiangular tight frames (ETF) resulting in bounded objective error. This encoding allows the nodes to solve the problem as if it were the original, eliminating the need for careful cloning or replication that requires maintaining the same state. The redundancy is directly embedded in the formulation of the optimization problem, altering the problem itself. There are several advantages to this approach. First, the nodes can operate without any knowledge of the encoding, making it compatible with existing distributed computing infrastructure and software. Second, with proper encoding, this method converges deterministically to an approximate solution to the original problem, even with stragglers, requiring just a small amount of redundancy. This is often sufficient for many machine learning and data analysis tasks that prioritize good generalization error over exact optimality.

In the domain of serverless systems, where tackling large-scale convex optimization problems presents challenges, [31] proposes OverSketched Newton. This algorithm stands as a randomized Hessian-based optimization approach tailored for such computational tasks within serverless architectures. By harnessing matrix sketching techniques, OverSketched Newton demonstrates remarkable resilience against stragglers.

A heterogeneous coded matrix multiplication (HCMM) algorithm is proposed in [32] for distributed matrix-vector multiplication, considering shifted exponential and shifted Weibull machine run-time distributions. It tackles the problem of minimizing average run-time with a tractable alternative formulation, and the proposed HCMM load allocation scheme is asymptotically optimal.

The computation-communication load region for linear output functions is fully characterized in [33], showing that it is primarily influenced by a single corner point, which achieves the minimal computation load while maximizing spectral efficiency. To do so, edge nodes should perform coded computations on linear combinations of inputs instead of individual inputs. A Universal Coded Edge Computing (UCEC) scheme is proposed for linear functions in MEC, minimizing computation load and maximizing physical-layer communication efficiency. Edge nodes generate coded inputs and compute coded output results to create communication messages that zero-force interference signals for each user. The scheme is universal, as coded computations are oblivious to channel states during communication.

To mitigate stragglers in distributed optimization, [34] proposes the method that embeds redundancy directly into the data itself, enabling computations to proceed without any awareness of the encoding process. They introduce multiple encoding strategies and demonstrate that popular batch algorithms, such as gradient descent and L-BFGS, can be implemented in a coding-oblivious manner to achieve deterministic sample path linear convergence to an approximate solution of the original problem. This convergence occurs using a varying subset of nodes at each iteration, and the approximation can be adjusted based on the level of redundancy and the number of nodes utilized in each iteration.

A good encoding matrix S should fulfill several requirements. Firstly, it needs some redundancy in its encoding vectors (the rows s of S). Secondly, based on the channel coding theorem, each encoding vector should provide as much independent information as possible. Lastly, the encoding matrix should not introduce errors, meaning that the exact solution can be recovered in the absence of failures, assuming nodes are unaware of the coding.

The primary goal of designing codes is to ensure that the degradation of solution accuracy is as gradual as possible as the number of failed nodes increases. ETF are appealing coding vectors due to their intrinsic redundancy, the independent information provided by each element, and the ability to reconstruct the exact solution without node failures, as observed numerically and analytically. Redundancy techniques for straggler mitigation in distributed optimization and learning are further studied in [35].

A new approach to solving inverse problems in distributed computing settings, called coded distributed computing, is proposed in [36]. Inverse problems are common in many fields, including imaging, signal processing, and machine learning, and involve estimating the underlying parameters of a system from noisy observations. Coded distributed computing involves using coding theory to design efficient and fault-tolerant distributed algorithms for solving inverse problems. The approach involves encoding the data and the computation and distributing the encoded data across multiple computing nodes, which can then perform computations locally and communicate with each other to collectively solve the inverse problem.

The authors demonstrate the effectiveness of coded distributed computing on several inverse problems, including image denoising, image deblurring, and compressed sensing. They show that the approach can achieve significant speedup and scalability compared to traditional centralized computing methods, and can also robustly handle failures and communication errors. The authors also analyze the theoretical properties of the approach, showing that it can achieve the optimal trade-off between communication and computation costs and that it can be applied to a wide range of inverse problems with different problem structures and noise models.

C. OTHER CODING SCHEMES

Some other coding schemes are worth mentioning in coded DML. Generalized PolyDot, a coding technique weaving coding into completely decentralized large Deep Neural Networks (DNNs) to deal with soft errors such as bit flips, is proposed by [37]. It provides unbounded better error tolerance.

The challenge of transmitting correlated sources to multiple receivers through a network of point-to-point channels with noise is studied in [38]. They introduce a new code construction technique called Neural Network Coding (NNC) as a solution. The NNC approach offers several key advantages, including:

- its reliance on a seed dataset of examples rather than assumptions about source statistics;
- its ability to function as an end-to-end communication scheme, i.e., a joint source and network coding scheme;
- the availability of practical decoders;
- its applicability to all network topologies;
- its compatibility with a range of power constraints at both the source and intermediate nodes.

The NNC approach employs neural networks (NNs) as encoders at the source and intermediate nodes and decoders at the destination nodes. The construction of the network code is jointly designed with the encoding and decoding phases of the transmission, converting real-valued input signals into channel inputs and then reversing the process to reconstruct real-valued signals from channel outputs. The NNC scheme can undergo offline training and testing on vast datasets to achieve optimal performance and remains highly feasible for implementation. These codes can achieve the power-distortion trade-off.

In distributed learning, model aggregation can be viewed as an advanced form of the indirect multi-terminal source coding problem, commonly referred to as the CEO problem. A distributed source coding (DSC) scheme is proposed in [39], called successive Wyner-Ziv coding, to exploit the information redundancy of workers in distributed learning. To reduce communication latency, the idea of low-precision training has been considered. For example, data is compressed by quantization here. Also, Slepian-Wolf (SW) coding is implemented by LDPC code.

A robust distributed computing framework is introduced in [40], which is capable of both approximate and exact

TABLE 3. Key findings in Section III Coded Edge Computing in DML.

Coding Techniques	Content	References
Coded Edge Computing in DML	Distributed workers function as edge nodes, performing edge computations	[1] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54]
Diversity Gain	Use straggling diversity similar to diversity channel fading in random fading channels	[55] [56] [57] [58] [59] [60]
Learning-based Coded Computation	Use the learning-based framework to generates codes	[61] [62] [63]
Coding for Cloud ML Systems	Coding methods for cloud ML with a huge amount of parameters	[64] [65] [66] [67] [68]
Privacy and Security in DML	Safeguard the confidentiality and security of datasets in DML tasks	[69] [70] [71] [72] [73]

computation of linear operations, particularly resistant to stragglers. The proposed approach combines polar codes and randomized Hadamard sketches in the DML system with the erasure channel, offering anytime estimation, flexibility, efficiency, and fast decoding for large-scale linear operation computations.

III. CODED EDGE COMPUTING IN DML

Coded edge computing [1] has gained increasing attention in DML, where individual distributed workers function as edge nodes, executing computations at the edge of the network. In [1], the advantage of applying coded edge computing in federated multi-task learning (FMTL) [43] is demonstrated, exhibiting a lower mean-squared error (MSE) and a slower rate of MSE increase as the error probability increases, in comparison to MDS-based coding schemes and uncoded methods. In this section, we research coded edge computing techniques used in DML and investigate the gains achieved, particularly in terms of reducing latency and enhancing accuracy. Table 3 provides a summary of the key findings from the researched references regarding coding in DML.

The Lagrange Coded Computing (LCC) framework [44] addresses challenges inherent in distributed learning algorithms, particularly those involving computations over large datasets distributed across multiple workers. LCC aims to provide resiliency against stragglers, security against Byzantine workers, and information-theoretic privacy of the dataset in a distributed setting. It achieves this by leveraging Lagrange polynomials to create computation redundancy in a novel coded form across workers, extending beyond linear computations to cover arbitrary multivariate polynomial functions. LCC demonstrates improved computation and communication efficiency and achieves an optimal tradeoff between resiliency, security, and privacy. Table 4 presents a comparison of the total runtime of the GC, LCC [44], matrix-vector multiplication based (MVM) approach [6], and uncoded schemes when applied to linear regression, highlighting the advantage of coded schemes. In each scenario, the tuple (m, d) denotes the feature matrix comprising m data points and d features. It is evident that LCC consistently exhibits the shortest runtime across all scenarios.

Coding techniques within the context of distributed fog computing are introduced in [45]. They propose two key coding schemes: minimum bandwidth codes and minimum latency codes. The minimum bandwidth codes aim to exchange computation redundancy at edge nodes for bandwidth optimization, while the minimum latency codes focus on mitigating the impact of stragglers to reduce latency and enhance system robustness. Additionally, they propose a unified coding framework that integrates both schemes, acknowledging the inherent tradeoff between communication load and computation latency.

TABLE 4. Comparison of the total runtime in seconds of GC, LCC [44] and uncoded schemes.

Schemes	Scenario 1 (8000,7000) Naturally occurring stragglers	Scenario 2 (8000,7000) Artificially introduce stragglers	Scenario 3 (160000,500) Artificially introduce stragglers
Uncoded	24.2	52.7	42
GC	8.2	16.7	11.5
LCC	3.4	3.77	4.4
MVM	3.4	5.4	57.5

A hybrid coding scheme that combines elements from both uncoded and MDS schemes is introduced in [46]. This hybrid approach merges the resilience against stragglers provided by MDS coding with the beneficial downlink transmission cooperation of the uncoded method. The study also demonstrated, from an information-theoretic perspective, that by controlling the robustness to stragglers and cooperation opportunities through specific parameter design, communication latency can be reduced at the expense of increased computation latency. The hybrid coding scheme outperforms uncoded situation and MDS method.

The focus of [47] is on improving the robustness of vehicular fog networks (VeFNs), particularly in scenarios characterized by unstable links between cars. They propose a scheme leveraging redundancy through coding techniques to mitigate the impact of link instability, thereby reducing the overhead of computation offloading. To determine the optimal coding scheme, they utilize a code-based computation offloading algorithm based on simulated annealing (CBSA).

The predictable and unpredictable mobility of nodes in the Internet of Mobile Things (IoMTs) is studied in [48]. In large networks, encoding and decoding procedures of fog computing become complex and costly. So, it proposes the utilization of Fountain codes [74], a flexible and uncomplicated coding scheme, to reduce the latency. Operating within the MapReduce framework [75], they examined three distinct relative positions among nodes. As a result, in addition to the conventional Map, Data Shuffling, and Reduce stages, they introduce an Output Delivery stage to accommodate the unique characteristics of node mobility in IoMTs.

In [49], the focus shifts from addressing straggling workers to tackling straggling links between helpers and clients within a distributed learning framework. The paper proposes the utilization of repetition coding (RC) and MDS coding to reduce the communication load, all without the presence of a central master node. The total communication load consists of clients-helpers and helpers-master communication load. With RC, the coded message is simply repeated, while MDS employs a Vandermonde construction. Furthermore, it characterizes the communication loads of both uplink and downlink channels, highlighting the tradeoffs between them. This analysis aids in determining the optimal number of helpers, clients, and masters required for deployment, along with the appropriate selection of uplink and downlink communication rates.

In [50], the focus is on reducing response time and improving resource efficiency in heterogeneous systems, where workers have varied computing and communicating capabilities. The paper proposes a joint scheduling-coding scheme to assign tasks based on their coded computational load. Redundancy is introduced by coding and tasks are distributed with the idea of load balancing and queuing theory. By assigning tasks that need large computation loads to workers with large computation resources and making each worker have a similar finishing time by load balancing, the approach maximizes utilization for heterogeneous workers and reduces the latency.

Unlike conventional transmission, where batch data is divided into non-overlapping mini-batches and offloaded to individual edge nodes (EN), coding schemes introduce redundancy at the expense of performance. To avoid transmitting redundant data, [51] proposes a novel coding-aware rate splitting scheme that initiates by extracting exclusive mini-batches. These mini-batches are then intelligently split using coding-aware techniques, facilitating multicast transmission. To lower the latency in distributed edge learning, a concave-convex procedure (CCCP) framework-based iterative optimization algorithm to solve the problem that has been converted to a difference of convex form by using auxiliary variables.

In [52], optimal incentive mechanisms for workers in coded ML are studied, specifically examining the trade-off between platform time cost and payment in the worker's payoff function. This mechanism takes into account workers' multidimensional heterogeneity in computation performances and costs. To simplify the complex worker selection problem, they summarize their heterogeneity into a one-dimensional metric and solve it with linear complexity. According to their research, if the platform has limited knowledge of workers' costs, it is best to assign loads based solely on their computation performances. In cases where the platform lacks insights into workers' computation performance, it is optimal to design cost- and performance-dependent loads.

In [53], the focus is on distributed multi-task learning (MTL) with asymmetric data placement where conventional coding schemes are not applicable. It also distinguishes itself from gradient coding, which revolves around a single task and linear operations on local gradients for global updates. In contrast, multi-task learning involves non-linear operations. The study introduces innovative coded computing schemes tailored for flexible and fixed data placements to mitigate uplink and downlink communication loads. By leveraging workers' local information and applying coding techniques, these schemes significantly reduce communication overhead. The paper establishes information-theoretic lower bounds on optimal communication loads, demonstrating the effectiveness of the proposed schemes. Additionally, it gives a communication-computation tradeoff, offering valuable insights for resource allocation.

Shifting from designing the coding scheme, [54] focuses on how to appropriately split and allocate work to workers, that is, setting the number of workers N and K needed for subtask result. The paper first models the problem as a Markov decision process and proposes a DRL-based workload allocation (DWA) algorithm to learn the strategy. The overall computation time of his method outperforms other methods.

A. DIVERSITY GAIN

The distributed computing framework exhibits similarities to random fading channels. This analogy implies that coding techniques can effectively capitalize on straggling diversity, just like how coding is utilized in communication channels to convert diverse channel fading into a beneficial factor. What makes coding particularly appealing in this context is its ability to harness diversity gain, which surpasses that achievable through mere replication in fading channels:

- diversity/parallelism trade-off in distributed systems with redundancy;
- the diversity versus parallelism tradeoff.

Distributed computing is necessary for the linear inverse problem of the system matrix due to the large memory needed. In [55], sparse generator matrices, low-density generator matrices (LDGM), are used because the traditional MDS codes will make the sparse matrix into a dense one which introduces extra storage cost and higher computation time. Although the poor ability of LDFM in error-correcting, to maximize the use of partial coded results in the power iteration computation and fill in missing information with

available side information, it proposes a new "substitute decoding" algorithm that can nearly achieve the convergence rate of noiseless power iterations. The larger the rank of the (partial) generator matrix, the more error would be reduced. In [56], it further gives applications of substitute decoding, including computing eigenvectors, the truncated singular value decomposition (SVD), and gradient descent.

Unlike the traditional method where long dot products are required, [57] considers short-dot products for the problem of computing a matrix-vector product Ax using distributed processing nodes with stragglers. In short-Dot, only a fraction of x is accessed by each processing node and redundancy is introduced in computation, which is helpful when communication is constrained. This novel error-correcting coding scheme tolerates stragglers and detects and corrects erroneous computations, making it applicable to computing systems with faulty hardware. It shows that Short-Dot has lower expected computation times, and derives fundamental limits on the trade-off between the length of the dot products and the recovery threshold, where Short-Dot is near-optimal.

Regarding stragglers in distributed matrix computations, rather than the generally considered block codes, [58] introduces a convolutional coding approach that is optimal in straggler resilience and numerically robust, as long as worker storage capacity exceeds the lower bound. It proposes a second approach with slightly higher decoding complexity but allows for greater operation flexibility near the lower storage capacity limit. Future work could explore other classes of convolutional codes, and consider both row and column-wise decomposition. Additionally, it investigates whether convolutional codes can efficiently handle errors, rather than just erasures.

In [59], they use a third-party trusted node as a helper node to detect malicious behavior in a lossy wireless communication channel, that is, the watchdog-based approach to resist pollution attacks. VANDER, the proposed scheme, leverages the properties of the Vandermonde matrix, of which each column subspace is calculated efficiently, and uses finite-field linear operations at the watchdog to effectively detect pollution attacks and ensure security while achieving both low false alarm and misdetection probabilities. It also shows the tradeoff between overhead and misdetection/false alarm probabilities.

In [60], a comprehensive investigation into distributed matrix multiplication with stragglers is conducted, introducing two novel coding schemes: Factored Luby Transform (FLT) and Factored Raptor (FRT) codes, adapted from Luby Transform (LT) and Raptor codes, respectively. It focuses on the optimality of FLT codes for probabilistic recovery threshold under peeling decoding, especially in scenarios involving random node failures. The paper studies the asymptotic regime of FLT codes, affirming their efficacy through rigorous analysis. Notably, the accuracy of density evolution equations in predicting error probability is proved. It shows that randomly sampled FLT codes, when subjected to peeling decoding, closely match ensemble average error performance. Using density evolution recursion formulas, the (probabilistic) recovery threshold of FLT codes for the Soliton degree distribution is computed. This threshold tends to approach the optimal recovery threshold as the degree of polynomials grows.

B. LEARNING-BASED CODED COMPUTATION

Research on learning-based coded computation presents a promising direction for optimizing distributed computation systems, offering a powerful combination of coding theory and machine learning techniques.

In [61], a novel approach for optimizing distributed computation systems is proposed, combining coding theory with machine learning techniques. The authors introduce a learning-based framework that generates codes tailored to the specific requirements of distributed computation, thereby reducing communication costs and improving overall computation time. The proposed approach employs neural networks to generate codes that adapt to different system settings and data distributions, ensuring efficient computation across various scenarios.

In [62], a learning-based coded computation framework is proposed to address the limitations of existing approaches in supporting general non-linear computations. The framework leverages machine learning to extend the applicability of coded computation to tasks involving non-linear functions, particularly focusing on neural network inference. By employing two paradigms: learning a code and learning a parity computation, the study demonstrates the effectiveness of learning-based approaches in accurately reconstructing unavailable results and reducing tail latency in distributed systems. Unlike traditional coding techniques, the proposed learning-based approaches require retraining learned components for each new computation encountered.

While LCC effectively solves matrix polynomial problems in a distributed coded way, it has limitations in solving general functions. To address this, [63] proposes AI-aided coded computation (AICC), an AI-aided learning approach inspired by LCC but incorporating DNNs for more general coded computation. AICC uses a system design similar to LCC, recovering through interpolation, but with a fixed recovery threshold independent of inputs or polynomial degree. Despite providing only approximate solutions, AICC shows sufficient accuracy for many practical applications, offering a wide range of potential uses. Experimental results illustrate the effectiveness of the proposed approach, showcasing significant improvements in communication cost and computation time compared to existing methods. The authors validate the applicability of their approach on real-world datasets, underscoring its potential for practical deployment.

C. CLOUD ML SYSTEMS

As the scale of machine learning datasets and the complexity of state-of-the-art DNNs continue to grow, there is a growing interest in effective distributed training methods. Modern

NLP models, such as BERT [76] and RoBERTa [77], possess hundreds of millions of parameters and often require days or even weeks of training time, particularly without access to large GPU clusters. However, utilizing these clusters presents significant systemic challenges, primarily concerning communication among GPUs. Inefficient designs can result in reduced scaling efficiency for large models, leading to the wastage of computational resources. Initially, the predominant approach to distributed training was parameter-server-based. This method involves partitioning model parameters across multiple parameter servers. Worker nodes would then sample a batch of data, retrieve the latest model parameters from the servers, compute gradients on the sampled data, and transmit the results back to the servers. This process can occur either synchronously, where each worker node waits for updates from all other nodes before proceeding, or asynchronously, where worker nodes independently fetch parameters and push gradients. Initially, parameter-server-based methods were widespread in distributed training, but scalability challenges prompted a shift towards all-reduce-based techniques. However, recent advancements in cloud networking technologies, including the elastic fabric adapter (EFA) and scalable reliable datagram (SRD), have reignited interest in reevaluating the parameter server approach.

The data is distributed across multiple workers, and redundancy is introduced using codes. Data sent by the fastest k out of n workers is sufficient to decode the computation. Addressing the challenge of dynamic computational environments where worker capabilities and cloud conditions are unknown, [64] presents a dynamic computational model called context-aware online coded computing. This model, formulated as a combinatorial-contextual multi-armed bandit (CC-MAB) problem, considers factors such as computation load and task output size that impact results. It is proven to be asymptotically optimal.

The basic ECCs implemented in the scheme proposed in [65] are extended Cauchy (EC) codes. The study presents an efficient decoding algorithm for local and global hierarchical codes, particularly focusing on EC codes. This scheme works for any Cauchy-like codes. It enables the application of hierarchical codes in computational storage and demonstrates their superiority over existing codes in terms of scalability and flexibility.

Modern cloud data storage systems face increasing demands for scalability, flexibility, and heterogeneity to accommodate growing data from various sources and dynamic usage rates. While shorter codes offer lower latency, they may lack the ability to effectively deal with erasures. Codes with hierarchical locality have emerged as a promising solution to reduce reading time while maintaining overall robust erasure correction capabilities. In [66], the first codes with hierarchical locality are designed to achieve scalability and flexibility in heterogeneous cloud storage environments, even with small field sizes. The innovative approach involves a double-level construction employing Cauchy Reed-Solomon codes over a finite field with a linearly growing size determined by the maximum local code length. Additionally, the triple-level construction exhibits scalability suitable for generalization across various hierarchical structures.

Modern cloud data storage systems face challenges due to the increasing volume of data from diverse sources and fluctuating usage rates. To address these challenges, [67] proposes a joint coding scheme for decentralized storage networks (DSNs) that combines hierarchical locality with topological properties. This scheme provides additional protection to nodes by leveraging cooperation among neighboring nodes in heterogeneous DSNs with any given topology. The hierarchical coding scheme improves reliability because information flows from more reliable nodes to less reliable ones. The proposed construction preserves scalability, flexibility, and adaptability to arbitrary topologies, critical for dynamic networks like DSNs.

To utilize the redundancy among those reusing matrices across batch jobs, inspired by Cross-Subspace Alignment codes, [68] introduces the Variable Coded Distributed Batch Matrix Multiplication (VCDBMM) problem and proposes Flexible Cross-Subspace Alignments (FCSA) codes as a solution with adjustable parameters for system complexity and high straggler resilience. It establishes a recovery threshold for VCDBMM and demonstrates the optimality gap is a constant factor of 2 for certain FCSA codes. Simulation results show that FCSA codes outperform other coding schemes. Future work includes reducing the optimality gap and extending FCSA codes to other computation tasks.

D. PRIVACY AND SECURITY IN DML

The aggregation of sensitive information from multiple sources and the data exchange and computational collaboration inherent in DML introduce significant privacy breaches and security risks. Therefore, implementing robust privacy- and security-enhancing techniques is crucial to safeguard the confidentiality and security of individuals participating in DML tasks. In this section, we research the coding techniques that aim to enhance privacy and security in DML. In Section IV-C, we will further explore coding techniques that aim to bolster privacy and security in FL, encompassing encryption, Shamir's secret sharing, Lagrange coded computing, and the code-based McEliece cryptosystem, among others.

To protect the privacy and security of DML, [69] proposes CodedPrivateML to keep the confidentiality of the dataset even if T out of the N machines that the dataset offloads data to collude. Specifically, the coding process is divided into two steps. Stochastic quantization transfers the dataset and the weight vector from a finite field to the real domain. Then the quantized values are encoded using Lagrange coding. To tackle the problem of using linear Lagrange to represent non-linear logistic regression, CodedPrivateML utilizes non-linear sigmoid to polynomial approximate it. Workers compute the assigned

TABLE 5. Comparison of the runtime and accuracy of AVCC [73], LCC, and uncoded schemes with varting stragglers and Byzantine nodes.

Schemes	Compute Time(s)	Communication Time(s)	Verification Time(s)	Decoding Time(s)	Accuracy
			$S = 0, M = 0$		
Uncoded	3.79	4.54	\	\	1
LCC	3.77	4.74	0.02	\	1
AVCC	3.78	4.88	0.007	0.042	1
			$S = 1, M = 2$		
Uncoded	6.19	4.52	\	\	1
LCC	3.77	4.75	0.02	\	93.7%
AVCC	3.42	4.93	0.003	0.045	95.1%
			$S = 2, M = 1$		
Uncoded	6.16	4.50	\	\	1
LCC	4.43	4.82	0.02	\	95.1%
AVCC	3.39	4.93	0.004	0.048	95.1%

coded data with the same computing structure as the uncoded one. The master decodes the returned gradients of several fast workers in the finite field and converts them to the real field to upgrade the weights. An information theory perspective threshold on privacy is given. Simulations on the performance of latency are also conducted to show the speedup.

Considering the scenarios of secure coded distributed learning in mobile Internet of Things (IoT) networks, [70] proposes secure distributed matrix multiplication (SDMM) for information-theoretic security and algorithms to reduce running time and save energy. There are four phases:

1) data partitioning; Data is partitioned according to the number of workers;
2) data encoding; Partitioned data is encoded by a random matrix for privacy, so in the information-theoretic point of view, no origin information can be obtained;
3) distributed computing; The data generated after encoding and partitioning are distributed following SDMM and the computations are conducted locally;
4) data decoding; By gradient descent in [69], weighted computed gradients are revealed.

In [71], a coded state machine (CSM) consisting of coded state and coded execution is introduced to tackle the influence of Byzantine faults. When behaviors deviate from the normal consensus phase and execution phase, they would be detected. As the network size gets larger, the storage efficiency and security improve linearly.

Traditionally, differentially private DML endeavors to protect privacy by introducing artificial noise into computing outcomes. However, in a novel scheme proposed for coded DML over Gaussian multiple-access wireless channels [72], the natural noise inherent in the wireless environment is leveraged to not only enhance energy efficiency but also ensure privacy preservation. This innovative approach utilizes differentially private Lagrange encoding to transmit local computing results to the master node via orthogonal channels. The scheme establishes achievable privacy protection levels to evaluate the impact of transmit power and power allocation on privacy and derives theoretical upper bounds for convergence, providing insights into system limitations and capabilities.

In [73], a further enhancement of LCC, known as adaptive verifiable coded computing (AVCC), is proposed to address issues related to stragglers, privacy protection, and mitigation of Byzantine workers. AVCC distributes coded data to N workers and decodes the results returned by the fastest K workers. While AVCC utilizes LCC for straggler tolerance and ensuring privacy, it employs verifiable computing to detect Byzantine workers. Unlike LCC, AVCC's verification process can be conducted independently for each worker node, eliminating the need to wait for results from other workers. Consequently, AVCC can operate with fewer workers compared to LCC. Secondly, AVCC implements a dynamic coding approach, automatically adjusting the trade-off between straggler tolerance and Byzantine protection. To tolerate S stragglers and M Byzantine workers, the iteration cost of compute time, communication time, verification time, and decoding time as well as the accuracy of the uncoded scheme, LCC and AVCC [73] using GISETTE dataset are compared in Table 5. Although AVCC incurs additional latency due to decoding and verification processes, when stragglers are present within the cluster, the overhead associated with decoding and verification in AVCC becomes negligible compared to the latency caused by stragglers. The presence of stragglers leads to a significant increase in uncoded execution time. Nonetheless, both LCC and AVCC demonstrate the capability to tolerate stragglers and Byzantine nodes, with AVCC achieving superior test accuracy.

Now, we summarize the trade-offs, including privacy and accuracy, privacy and communication load, accuracy loss due to quantization, and increased system complexity, etc.

TABLE 6. Key findings in Section IV in Federated Learning.

Coding Techniques	Content	References
Coding for Straggling	Coded Computing	[5] [79] [80] [81] [82]
	Coded Gradient Aggregation	[15] [83] [84] [85]
Communication Loads	Minimize Communication Overhead by Optimizing the Transmission of Information	[86] [87] [88] [89]
Privacy and Security in FL	Secret Sharing & Lagrange Coding & Code-based Cryptosystem	[90] [91] [92] [93]

In [73], the trade-off exists between the straggler tolerance and Byzantine protection. In the setting of [72], experimental findings reveal a trade-off between system resource settings, convergence, and privacy protection levels. Notably, increasing SNR and power allocation for gradient computation leads to decreased privacy protection levels but improved training accuracy and fewer dataset partitions result in enhanced training accuracy.

Then, taking CodedPrivateML in [69] as an example, a trade-off exists between privacy and the computational load of each worker. This trade-off arises from the allocation of additional workers, which can either defend against collusions or enhance parallelization. The computational load at each worker node is inversely linked to the degree of parallelization. It is proved that with an increase in the number of workers within a cluster, both the degree of parallelization and the privacy threshold of CodedPrivateML can linearly increase. Furthermore, to ensure information-theoretic privacy, [69] employs quantization to transform both the dataset and model into the finite field $\mathbb{F}_p$. Accuracy is traded off for privacy, especially with fewer iterations. As the number of iterations grows, particularly reaching 50 iterations, CodedPrivateML achieves nearly identical accuracies to conventional logistic regression (utilizing the sigmoid function without polynomial approximation or quantization), while still maintaining privacy. The potential performance degradation resulting from quantization can be alleviated by employing floating-point numbers instead of fixed-point numbers.

Increasing the number of workers decreases the computation load on each worker, but it also leads to an increase in the encoding time at the master node. Consequently, while adding more workers initially provides benefits, there comes a point where the gains become marginal, and the system may saturate. In such cases, adding more workers may no longer reduce the training time, as the encoding overhead begins to dominate the computation process.

IV. CODING IN FEDERATED LEARNING

Federated learning involves training machine learning models on multiple edge devices that are distributed across a network while keeping the data decentralized and local. Although this approach can help to reduce the resource requirements and protect privacy for training large-scale models, it also introduces new challenges, such as communication delays, data loss and further privacy protection requirements. In [78], open problems in the heating FL structures, such as improving the efficiency and effectiveness, preserving the privacy of user data and defending against attacks and failures, are studied. Coding techniques have been considered promising ways to address these challenges. In this section, we mainly discuss the application of coding techniques to solve problems in federated learning, including straggling, communication loads and privacy problems. Table 6 lists the key findings from the researched references regarding coding in FL.

A. STRAGGLING

The heterogeneity of computing power among edge devices and variations in network communication quality often limit the training efficiency of federated learning, leading to straggling issues. Error correction codes are a common coding technique used to address this challenge. They can detect and correct errors in data generated by edge devices and transmitted over networks. By adding redundant data to the original data, error correction codes enable the detection and recovery of data in the presence of errors. Applying error correction in federated learning helps manage network and node failures, as well as compensate for communication delays caused by stragglers.

1) CODED COMPUTING

Coded computing introduces redundancy into the computation and communication process, allowing for efficient recovery from errors and improving overall system performance. By encoding data and computations using error-correcting codes, coded computing can mitigate the impact of stragglers and reduce the amount of data that needs to be communicated.

To mitigate the impact of stragglers, [5] proposes a novel coded computing scheme for federated learning called Coded Federated Learning (CFL) as Figure 2 illustrated, designed for linear regression models. Each client generates local parity data utilizing random linear coding before training, which then be sent to the central server. After performing local updates, each client uploads its local model parameters to the central server. Instead of waiting for all straggling parameters, the central server sets a deadline and computes parity gradients over global parity data to compensate for the straggling nodes and links. By combining these parity gradients with the aggregation of the received gradients, the central server can obtain an overall global model.

TABLE 7. Convergence time ($\times 10^4$ s) for different δ and NMSE in CFL [5].

NMSE	Coding Redundancy δ					
	0 (Uncoded)	0.04	0.1	0.16	0.22	0.28
0.1	1.20	1.37	1.64	2.02	2.53	3.10
0.01	2.64	2.45	2.38	2.48	2.90	3.41
0.001	4.30	3.70	3.27	3.05	3.07	3.79

TABLE 8. Comparison of coding gain and communication overhead for different coding redundancy δ in CFL [5].

Metrics	Coding Redundancy δ						
	0	0.04	0.1	0.16	0.22	0.25	0.28
Coding Gain	1	1.65	1.83	2.51	2.28	2.04	1.86
Communication Load	1	1.15	1.45	1.80	1.98	2.21	2.41

Specifically, each device i possesses local data $D_i = \{(x_j^{(i)}, y_j^{(i)}) \mid j \in [n_i]\}$ consisting of l_i points with input data $x_j^{(i)}$ and its corresponding label $y_j^{(i)}$. Denote the dataset matrix $\mathbf{X}^{(i)} = (x_1^{(i)}, \ldots, x_{n_i}^{(i)})^\top$ and $\mathbf{y}^{(i)} = (y_1^{(i)}, \ldots, y_{n_i}^{(i)})^\top$. Each client i randomly generates a matrix $\mathbf{G}_i$ and performs $\tilde{\mathbf{X}}^{(i)} = \mathbf{G}_i\mathbf{W}_i\mathbf{X}^{(i)}$, $\tilde{\mathbf{y}}^{(i)} = \mathbf{G}_i\mathbf{W}_i\mathbf{y}^{(i)}$ on its training data set to obtain the parity data, where $\mathbf{W}_i$ is a $\ell_i \times \ell_i$ weight matrix. The diagonal coefficients of $\mathbf{W}_i$ corresponding to $k = 1, \ldots, l_i^*(t^*)$ data points are given by $w_{ik} = \sqrt{Pr\{T_i \geq t^*\}}$, where $Pr\{T_i \geq t^*\}$ is the probability that the central server does not receive partial gradient from the i-th device within epoch time t^*. Moreover, the remaining $(\ell_i - \ell_i^*(t^*))$ uncoded data points are set as $w_{ik} = 1$.

To measure the amount of coded data initially uploaded to the central server, they define the coding redundancy δ, where $\delta = 0$ corresponds to uncoded FL, and higher values of δ indicate more parity data being transmitted to the central server. The comparison in Table 7 shows the convergence time for various coding redundancy δ and normalized mean square errors (NMSE). For NMSE $= 0.1$, uncoded FL outperforms all coded schemes, as the transmission of parity data in CFL incurs additional communication costs. However, for NMSE $= 0.001$, the coded scheme with $\delta = 0.16$ achieves the minimum convergence time, indicating that CFL is effective when dealing with straggling problems. Furthermore, Table 8 highlights the trade-off between convergence rate and coding redundancy. While higher coding redundancy can better mitigate straggling issues, it also leads to increased communication costs due to the transmission of parity data, resulting in a reduction in communication gain. Overall, selecting an appropriate δ is crucial for optimizing performance.

In [5], CFL is constrained to linear regression models. To address this limitation, [79] introduces a novel framework called CodedFedL, enabling coded computing for non-linear federated learning tasks. This approach transforms non-linear training tasks into distributed linear regression problems using distributed kernel embedding via random Fourier features (RFF).

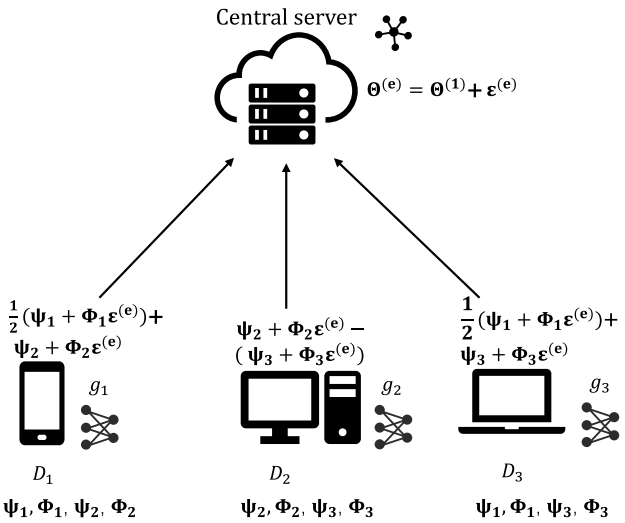


FIGURE 3. An example of CodedPaddedFL with three devices involves each device sharing one-time padded data Ψ_i and Φ_i with the other two devices. Subsequently, the devices send out coded gradients based on an ($\alpha = 2, n$) gradient code to the central server. The central server can decode these gradients upon receiving them from at least two devices, making the system resilient to one straggler.

In CodedFedL, edge devices locally generate masked parity data at the beginning of the training process by computing linear combinations of features and labels from their respective datasets. The central server then calculates the gradient based on the global parity data to derive the coded gradient. This coded gradient is combined with uncoded gradients to stochastically approximate the full gradient over the entire training dataset, thereby mitigating convergence issues arising from stragglers.

They utilize statistical methods to optimize the load distribution and coding redundancy settings based on the computing and communication costs of the clients. Additionally, they analyze the privacy of local parity data and the convergence performance of CodedFedL under specific assumptions. The numerical results demonstrate significant improvements over benchmark datasets. The server's computation performance, achieved by combining coded gradients from straggling clients with uncoded gradients from non-straggling clients, closely matches that of using uncoded gradients over the entire dataset.

However, sharing parity data with the central server compromises privacy, leading to a reduced level of privacy compared to conventional FL. To address this, [80] introduces CodedPaddedFL, which combines one-time padding for privacy with gradient codes to enhance resilience against stragglers while maintaining the privacy level of conventional FL. CodedPaddedFL is illustrated in Figure 3 and consists of the following two phases.

In the first phase, each device i will generate a pair of uniformly random one-time pads, denoted as $\mathbf{R}_i^{\mathrm{G}}$ and $\mathbf{R}_i^{\mathrm{X}}$, which are then sent to the central server to facilitate the use of gradient codes before training. These pads are used to encode the local gradient and data as follows: $\mathbf{\Psi}_i = \boldsymbol{G}_i^{(1)} + \mathbf{R}_i^{\mathrm{G}}$, $\mathbf{\Phi}_i =$

TABLE 9. Comparison of training time (h) for different coding schemes with training accuracy reaching 95% on MNIST and 85% on Fashion-MNIST [80].

Coding Schemes	Conventional FL	CodedPaddedFL(α) [80]				CodedFedL (δ) [79]	
		$\alpha = 6$	$\alpha = 16$	$\alpha = 23$	$\alpha = 25$	$\delta = 0.1$	$\delta = 0.8$
MNIST	1.8+	1.8+	1.00	0.78	0.79	1.8+	0.65
Fashion-MNIST	1.8+	1.8+	1.53	0.96	0.94	1.8+	0.83

TABLE 10. Comparison of test accuracy (%) with the same training time for different coding schemes [94].

Coding Schemes	FedAvg [4]	DP-CFL [95]	CodedFedL ($\sigma_i^2 = 0$) [79]	CodedFedL($\sigma_i^2 = 0.25$)	SCFL ($b_k^i = 32$)	SCFL [94]
MNIST	94.9	94.4	94.1	92.5	95.1	95.3
CIFAR-10	84.7	84.5	75.6	73.9	84.8	85.2

$\boldsymbol{X}^{(i)\top}\boldsymbol{X}^{(i)}+\mathbf{R}_i^{\mathrm{X}}$, where $\mathbf{X}^{(i)}$ denotes the dataset of edge device i and $\boldsymbol{G}_i^{(1)}$ represents the gradient of device i in the first epoch. Following this, devices share the padded matrices $\boldsymbol{\Psi}_i$ and $\boldsymbol{\Phi}_i$ with $\alpha-1$ other devices to introduce redundancy in the network. Each device then computes the gradient based on a linear combination of a subset of $\{\boldsymbol{\Psi}_1,\ldots,\boldsymbol{\Psi}_n\}$ and $\{\boldsymbol{\Phi}_1,\ldots,\boldsymbol{\Phi}_n\}$, where the linear combination is determined by an (α, n) gradient code, ensuring the system is resilient to $\alpha-1$ stragglers. In the subsequent phase, the devices and the central server collaboratively and iteratively train the global model $\boldsymbol{\Theta}^{(e)}$. As $\boldsymbol{\Theta}^{(e)}=\boldsymbol{\Theta}^{(1)}+\boldsymbol{\epsilon}^{(e)}$ at epoch e, the central server sends $\boldsymbol{\epsilon}^{(e)}$ instead of $\boldsymbol{\Theta}^{(e)}$ as in traditional federated learning. Upon receiving $\boldsymbol{\epsilon}^{(e)}$, devices compute the gradient on the encoded data.

CodedPaddedFL entails a significant communication overhead because of the need for data sharing and decoding at the central server to preserve data privacy. These requirements escalate with higher values of α. To minimize latency further, a grouping technique is proposed, which can uphold a high level of straggler mitigation even as α decreases. The approach involves dividing all devices into N equally sized disjoint groups, with each group executing CodedPaddedFL locally. The central server can then derive the global gradients by consolidating the aggregated gradients from all individual groups.

They simulate a federated learning network in which devices collaborate to train on the MNIST and Fashion-MNIST datasets. The comparison in Table 9 illustrates the training time for Conventional FL, CodedPaddedFL with varying values of α, and CodedFedL [79] with different coding redundancy levels δ, aiming to achieve a training accuracy of 95% on MNIST and 85% on Fashion-MNIST. The results demonstrate that CodedPaddedFL outperforms Conventional FL across all α values. Furthermore, it is evident that selecting the optimal α is crucial, as $\alpha=23$ achieves 95% accuracy on MNIST the fastest and $\alpha=25$ achieves 85% accuracy on Fashion-MNIST the fastest. In comparing CodedPaddedFL and CodedFedL, it is observed that CodedPaddedFL outperforms CodedFedL with $\delta=0.1$, although it is slightly slower than CodedFedL with $\delta=0.8$.

Stochastic Coded Federated Learning (SCFL) is a novel approach that combines concepts from stochastic gradient descent (SGD) and coded computing to enhance the efficiency and robustness of federated learning systems [81], [94]. In SCFL, each device participating in the FL process generates encoded data using random linear combinations and adds noise to the gradients before transmitting them to the central server. This encoding process helps mitigate the impact of stragglers and noisy communication channels, which are common challenges in FL. One key advantage of SCFL is its ability to improve training efficiency by allowing more training steps at both the server and the clients. By utilizing coded computations, SCFL enables the server to compute stochastic gradients continuously without incurring additional communication overhead. This approach enhances the convergence speed of the FL model. Moreover, SCFL ensures privacy protection by adding noise to the encoded data, which helps preserve the privacy of individual devices' data. The use of noise and encoding schemes also contributes to the overall robustness of the FL system against various types of attacks and failures.

They also conduct experiments to compare the proposed SCFL with FedAvg [4], CodedFedL [79] with a noise level of $\sigma_i^2=0.25$, and without adding noise, which is equivalent to $\sigma_i^2=0.25$. They also compare with DP-CFL [95], where only the server performs centralized model training using the global coded dataset without further cooperation with the edge devices. Table 10 illustrates the comparison of test accuracy with the same training time for different coding schemes. Here, SCFL ($b_k^i=32$) denotes SCFL with a constant batch size of $b_k^i=32$. It is noted that CodedFedL achieves the lowest test accuracy, primarily because of the frequent communication and model update bias introduced by the noisy coded datasets. Even in the absence of added noise (i.e., $\sigma_i^2=0$), the performance of CodedFedL remains inferior to other schemes. This further highlights the advantage of periodic averaging in hastening the convergence speed. DP-CFL achieves a similar test accuracy to FedAvg because the server continuously computes stochastic gradients without any communication overhead. SCFL achieves the highest test accuracy within the given training time compared to the baseline methods.

Table 11 illustrates the comparison of test accuracy among SCFL, DP-CFL, and CodedFedL under various privacy

TABLE 11. Comparison of test accuracy (%) with different privacy budget for different coding schemes [94].

Coding Schemes	Privacy Budget ϵ				
	6.5	7.0	7.5	8.0	8.5
SCFL [94]	94.5	95.1	95.4	95.5	95.5
DP-CFL [95]	89.4	91.1	92.5	93.5	94.6
CodedFedL [79]	81.0	82.6	86.0	88.8	92.4

budgets. SCFL consistently achieves the highest test accuracy across all privacy budgets, demonstrating its superior privacy-performance tradeoff. This is attributed to SCFL's ability to effectively counteract the adverse effects of added noise. The comparisons of various coding techniques in FL, including CFL [5], CodedFedL [79], CodedPaddedFL [80], and SCFL [81], are presented in Table 12.

An error correction coded FL system that prioritizes learning accuracy is proposed in [96]. Leveraging SGD, known for its resilience to errors during training, the scheme ensures minimal impact on the final learning accuracy. Operating under the framework of Managed Redundancy (FL-MR), the system is structured into two phases: the No-Retransmission (NR) phase, wherein decoding errors do not trigger retransmissions, and the Select Retransmission (SR) phase, which selectively retransmits lost gradients to minimize communication overhead. The efficient retransmission mechanisms also incorporate rateless coding for downlink communication and point-to-point ECC with Automatic Repeat reQuest (ARQ) for uplink, effectively addressing communication challenges in FL while upholding learning accuracy.

While most works on vertical federated learning focus on the theoretical studies related to accuracy and security, [97] concerns the overall consuming runtime caused by computation and communication during training. They introduces Coded Distributed Computing (CDC) by utilizing erasure coding techniques in matrix multiplication into vertical federated learning for regression (FLR) with the goal of speeding up the training process. They proposes two schemes, vertical FLR which is based on linear combination and Matdot-based vertical FLR, exploiting in-parallel computation and homomorphic encryption at each device. While LC-based FLR is superior considering computing time only, it involves more iterations of CDC and is susceptible to integrated encryption occupying a lot of time at the master node, which shows an advantage in small computation size at each worker node. Matdot entails CDC only one time and has the potential to distribute homomorphic encryption process across multiple parallel worker nodes to reduce the time spent significantly. Numerical results show that the introduced coded schemes are superior to conventional uncoded schemes markedly considering the time of encoding, computing, and decoding of the training process.

To speed up the training process of heterogeneous FL while retaining a certain level of security, [98] proposes a linear coded federated learning under multiple stragglers (LCFLMS) framework by designing a client-based multiple stragglers task outsourcing (C-MSTO) scheme and a server-based multiple stragglers task outsourcing (S-MSTO) scheme. Linear coding computing schemes will be applied to the process of outsourcing and the results reveal that LCFLMS scheme can speed up the training process.

While most existing CFL schemes are constrained by fixed coding redundancy parameters and the use of a weight matrix current introduces potentially unnecessary errors, [99] introduces a novel framework coded federated learning framework for the heterogeneous computing environment (CFL-HC). They consider a heterogeneous FL system consisting where parts of clients take raw data and parts of clients selected to receive coded data. SGD and a requested number of input-output pairs which is specified will be used in one round.

They obtain the optimization solutions of the best cut-off setting of each training round and the distribution load across clients. The numerical experiments are implemented on a real system using the message passing interface platform, which demonstrates the proposed framework achieves high accuracy and fast convergence at the same time.

To address the challenge of dealing with time-varying channel conditions and non-IID data distributions, [82] proposes a novel scheme that integrates FL with Slimmable Neural Networks (SNN), changing its width according to the heterogeneous energy capacity and model complexity. Integrating FL with SNN is difficult because existing multi-width SNN training algorithms are designed for standalone learning, thus the paper proposes a new FL framework called SlimFL, which aggregates global models by superposition coding (SC) and updates local models by superposition training (ST). Since the state-of-the-art SNN training technique called universal SNN (USTrain) fails to manage multiple batch normalization layers efficiently, the authors also propose a super positioned USTrain (SUSTrain) algorithm for clients to train the local model.

Specifically, there are two width configurations during the SNN training process, a half-width (0.5x) and a full-width model (1.0x), and the SNN architecture is separated into two segments, left-half (LH) and right-half (RH). After assigning different transmission power levels, each client transmits the encoding LH and RH to the server. The server only decodes the LH if the channel throughput is poor, generating a model with 0.5x, also the server may decode both LH and RH and merge them to produce the full-width model (1.0x) with a high-quality channel. Therefore, the server superimpose the decoded 0.5x and 1.0x local models to obtain a global model by Successive Decoding (SD) or Successive Interference Cancelation (SIC). Each client downloads the global model and trains the model until convergence. Overall, the process of SlimFL consists of four iterative steps:

- Superpositioned USTrain (SUSTrain);
- Superposition Coding (SC);

TABLE 12. Comparison of different key coding schemes in FL.

Coding Schemes	Content	Performance
CFL [5]	Generate parity data using random linear code and send it to the central server for compensation.	Trade-off between coding redundancy and communication gain. Outperform uncoded FL in training time but with low privacy level.
CodedFedL [79]	Enable CFL for non-linear regression using distributed kernel embedding with random Fourier features.	Outperform uncoded FL and greedy uncoded FL in training time for both IID and non-IID data.
CodedPaddedFL [80]	Share one-time padding data with other devices. Send coded gradients to the central server using gradient coding.	Enhance privacy level by using one-time padding. Outperform CodedFedL with specific coding redundancy in training time .
SCFL [81]	Add Gaussian noise to generate coded data. Enables periodical averaging with more local training steps.	Outperform CodedFedL in training time even with high noise level.

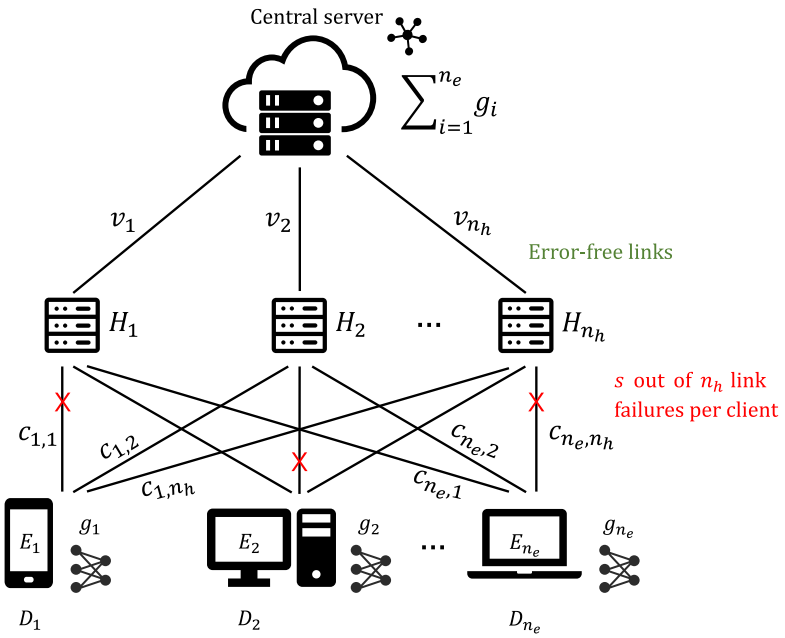


FIGURE 4. Hierarchical coded gradient aggregation. Each client node $i \in [n_e]$ encodes its local gradient to generate coded messages $c_{i,j}$ for the n_h helper links, with a maximum tolerance of s link failures. The central server utilizes the messages received from helpers to instruct the helpers on what to transmit, facilitating the recovery of the full gradient from all clients.

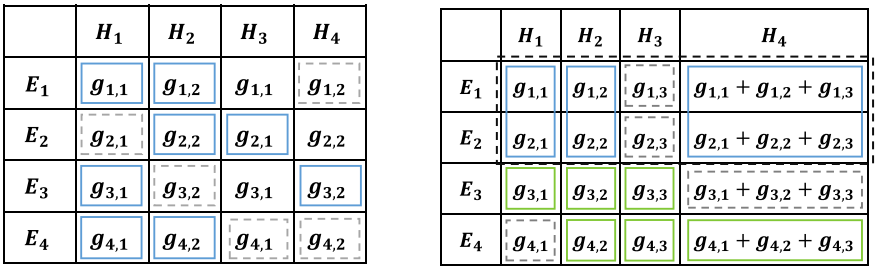
- Successive Decoding (SD);
- Aggregation.

Based on balanced Reed-Solomon (BRS) codes [100], [101] develops a coded matrix inversion (CMIM) for FL to mitigate the stragglers. The proposed scheme locally approximates the inverse without revealing the data to the server and utilizes MDS generator matrices to generate the erasure code. They utilize the same BRS generator matrix across the network, to linearly encode the local computations by gradient coding while each clients need to generate a random encoding matrix in [5].

2) CODED GRADIENT AGGREGATION

While the above proposed schemes mainly focus on coded computing by generating parity data to compensate for the straggling problems, [15] considers a coded gradient aggregation for learning at the edge. Previous researches on gradient coding [7] have primarily addressed centralized data placement while in FL the client data is private and generated locally, which makes a centralized data placement strategy impractical. They introduce coded redundancy in communication links for a hierarchical setup, where each of the n_e clients sends coded gradient updates to n_h helpers as Figure 4 illustrates. The aggregation process succeeds even if up to any s straggling links out of the n_h helper links per client. To achieve this, they propose the following two different hierarchical aggregation approaches and denote the normalized client-to-helpers communication load as C_{EH} and the normalized helpers-to-master communication load as C_{HM}.

- Aligned Repetition Coding. Aligned Repetition Coding (ARC) involves each client partitioning its gradient update and sending each unique component to multiple helpers. An intuitive example of ARC is illustrated in Figure 5(a) with $n_h = 4$, $n_e = 4$ and $s = 1$. While the communication links in grey boxes fail, each helper can still aggregate the data in blue boxes and send it to the central server to recover the full gradient. For example,



(a) ARC (b) AMC

FIGURE 5. Illustration of ARC and AMC with $n_e = 4$, $n_h = 4$ and $s = 1$. In Figure 5(a), grey boxes denote the message failures while the blue boxes denote the messages that helpers locally aggregate and then send to a central server. In Figure 5(b), the central server first finds the maximum number of exactly matching rows marked by a black box and the helpers locally aggregate the message from the blue boxes and simply forward the message from green boxes to the central server.

helper 1 obtains $g_{1,1} + g_{3,1} + g_{4,1}$ and helper 3 obtains $g_{2,1}$. By aggregating the data from helper 1 and helper 3, the central server can obtain $\sum_{i=1}^{4} \mathbf{g}_{i,1}$. This alignment ensures that corresponding gradient components from different clients align at the helpers, creating substantial opportunities for partial aggregation. This approach achieves a normalized helpers-to-master communication load C_{HM} of $O(n_h)$, serving as an upper bound for the optimal C_{HM}. However, ARC demands C_{EH} of $\Theta(n_h)$, which might be too high for communication-limited clients.

- Aligned MDS Coding. To address the issue of high C_{EH} in ARC, Aligned Minimum Distance Separable Coding (AMC) is proposed, which can achieve an optimal C_{EH} of $\Theta(1)$ for a specified resiliency threshold by applying an MDS code to the gradient components. This approach also achieves a C_{HM} of $O(n_e)$. AMC involves each client partitioning its gradient and applies an MDS code over the partitions. Figure 5(b) is a motivating example of AMC with $n_h = 4$, $n_e = 4$ and $s = 1$ by using a (4,3) MDS code, which means to recover any missing one of the 4 elements in a row requires all the remaining 3 elements in that row. The central server initially determines the maximum number of rows that have exact matches as the first and second rows in Figure 5(b) with a dotted black box. The data in blue boxes in the maximum matching rows are partially aggregated by the respective helpers and then sent to central server. The remaining data in green boxes are forwarded directly to a central server.

The scheme proposed including ARC and AMC in [15] demonstrates a trade-off between C_{EH} and C_{HM}. Building on this trade-off, [83] suggests a scheme that utilizes pyramid codes [102] to achieve a balance in communication costs between edge nodes and helper nodes. Pyramid codes are a class of erasure codes used in distributed data storage systems to achieve fault tolerance. In pyramid codes, the data is encoded into a pyramid-like structure, where each level of the pyramid represents a different level of redundancy. At the base of the pyramid, the original data is stored. As you move up the pyramid, redundancy is added hierarchically. This redundancy allows the system to recover the original data even if some of the encoded data is lost or corrupted. They provide a derivation of exact expressions for the communication costs at both edge nodes and helper nodes. This analysis provides insights into how the scheme performs under different scenarios and helps optimize system parameters for improved efficiency.

Since both AMC and the pyramid scheme revolve around a key aggregation strategy, which involves solving a combinatorial problem to determine the maximum number of rows in the codeword array that meet specific criteria, [84] further expands these strategies to μ-max AMC and μ-max pyramid schemes, respectively. Furthermore, they propose a new aggregation strategy to the ARC scheme by solving a set cover problem using greedy algorithm and the simulations indicates the greedy-based ARC outperforms the naive ARC [15].

Unlike traditional erasure codes that operate on individual bits or symbols, vector codes operate on entire vectors of symbols. This means that instead of encoding and decoding individual bits or symbols, vector codes encode and decode entire vectors, which can improve efficiency and reduce computational complexity. Vector codes at client nodes and layered short-block-length MDS codes to generate vector codes are introduced in [85]. The propose scheme allows for the attainment of intermediate operating points between the ARC and AMC schemes by balancing C_{HM} against C_{EH}.

B. COMMUNICATION LOADS

Coding is crucial in reducing communication overhead in FL. By employing coding techniques, FL systems can optimize the transmission of model updates and aggregated information, minimizing the amount of data exchanged between devices and the central server. This reduction in communication overhead improves the efficiency and scalability of FL systems, making them more practical for large-scale and resource-constrained environments. Coding techniques for reducing communication loads in this section include network coding and compression in FL.

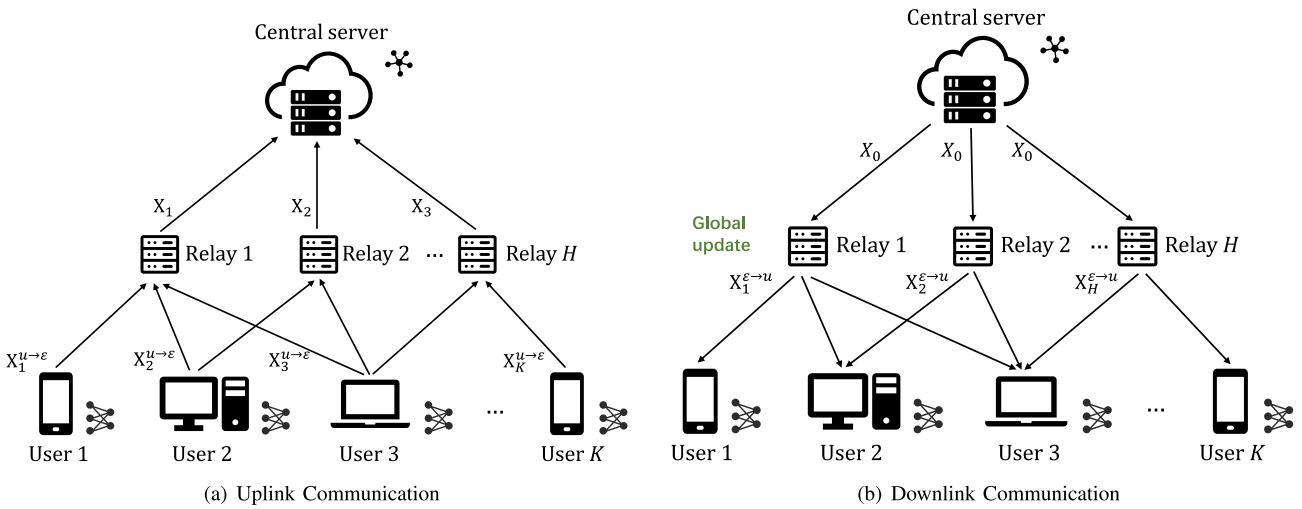


FIGURE 6. Illustration of coded hierarchical multi-task learning with K users jointly training multiple models $\{w_1, \ldots, w_K\}$ through H relays. During the uplink communication as shown in Figure 6(a), each user $k \in [K]$ performs local updates and sends message $X_k^{u\to\epsilon}$ to the connected relays. Each relay $i \in [H]$ then generates linear combinations of the received messages and sends X_i to the central server. During the downlink communication as shown in Figure 6(b), the central server sends the message X_0 to all relays, enabling each relay to perform global updates and send $X_k^{\epsilon\to u}$ to the users, thereby allowing each user $k \in [K]$ to obtain their own model w_k.

1) NETWORK CODING

Network coding is a technique to enhance data transmission efficiency and network robustness. Unlike traditional routing, which simply forwards packets along predefined paths, network coding allows intermediate nodes in the network to perform operations on the packets they receive. These operations can include combining multiple packets into a single packet or generating new packets based on the information contained in the received packets.

In contrast to previous coded computing methods that involve transmitting redundant data, the proposed coding scheme in [103] and [104] reduce the communication loads both in the uplink and downlink by leveraging the network topology for hierarchical distributed multi-task learning (MTL) including federated multi-task learning without sacrificing the convergence rate and learning performance.

They consider MTL framework which involves K users jointly training multiple models $\{w_1, \ldots, w_K\}$ with the assistance of a central server, ensuring that each user $k \in [K]$ obtains their desired model w_k after training. This process includes both uplink and downlink communication, as illustrated in Figure 6.

For uplink communication, to reduce communication costs by avoiding repeated messages, each client k splits its dataset into z_k disjoint segments, where z_k is the number of outlinks of client k. These segments are then sent to the connected relays according to the designed scheme. Each relay sends linear combinations of the received messages to the central server to recover the gradients. For downlink communication, the multicast gain is maximized by having relays simultaneously send updates to edge devices. The global update is moved to each relay, which already possesses certain messages from the uplink communication.

Table 13 compares the overall training time between the uncoded scheme and the proposed scheme with $H = 10$ relays and $H = 20$ relays. We observe that the proposed scheme with $H = 10$ relays outperforms the one with $H = 20$ relays under given user connectivity r, due to increased coding opportunities for each relay. Table 14 presents the comparison of overall training time between uncoded scheme and the propose scheme with user connectivity $r = 2$ and user connectivity $r = 3$ for different numbers of users. The proposed scheme outperforms the uncoded scheme. Furthermore, as the number of users and user connectivity increases, the proposed scheme becomes more efficient.

TABLE 13. Comparison of overall training time (s) for different coding schemes with different number of user connectivity [104].

Methods	The Average User Connectivity r					
	5	6	7	8	9	10
Coded MTL (H = 10)	10.7	9.4	8.1	6.7	5.1	3.1
Coded MTL (H = 20)	13.5	13.0	12.5	12.0	11.3	10.8
Uncoded MTL	14.5					

TABLE 14. Comparison of overall training time (s) for different coding schemes with different numbers of users [104].

Methods	The Number of Users K					
	15	20	25	30	35	40
Coded MTL (r = 2)	4.5	5.8	7.1	8.5	9.8	11.1
Coded MTL (r = 3)	3.9	5.1	6.4	7.5	8.7	10.0
Uncoded MTL	4.8	6.4	7.9	9.5	11.0	12.6

Based on network coding, [89] proposes Network Coding Federated Learning Systems (NC-FLSs). The proposed scheme considers the butterfly network with two clients and clients or central server will transmit the data by combining the divided two-part data. The central server decodes the original data after receiving enough independent combination messages. In [105], FedNC based on Random Network

TABLE 15. Comparison of test loss (nat) with the same training time for different coding schemes [86].

Coding Schemes	FedAvg [4]	QSGD [107]	STC [108]	FedPAQ [109]	UVeQFed [110]	Predictive coding FL [86]
Fashion-MNIST	0.33	0.42	0.39	0.22	0.19	0.13
CIFAR-10	0.40	0.56	0.67	0.97	1.04	1.60

Coding is proposed, which means sending random linear combinations of the original data to the sink nodes. The proposed scheme outperforms the privacy, efficiency, and robustness compared to traditional FL.

2) COMPRESSION

While quantization, a common compression technique applied to model weights or gradients, may compromise the accuracy of the model, combining it with entropy coding can help strike a balance between efficiency and data fidelity [86]. Entropy coding efficiently represents data by assigning shorter codes to more frequently occurring symbols and longer codes to less frequent ones [106]. It exploits the statistical properties of the data to achieve compression. In federated learning, entropy coding is employed to compress the gradients exchanged between the central server and the participating clients. By reducing the amount of data transmitted, entropy coding helps optimize communication efficiency and reduce the overall training time, especially in scenarios with limited bandwidth or communication resources.

To further improve the accuracy loss caused by direct quantization, [86] propose a predictive coding based compression scheme for federated learning together with entropy coding. Predictive coding works by predicting the value of each sample based on previous samples, and then encoding the difference between the predicted value and the actual value. By transmitting only the prediction error, which typically requires fewer bits than the original signal, predictive coding can achieve compression. The proposed scheme consists of prediction, quantization and entropy coding.

In the first phase, four distinct prediction modes are designed, and the mode with the least prediction error is chosen for optimal compression. This selected mode is then conveyed to the decoder through a dedicated two-bit mode information transmission. Then quantization is a process that maps continuous input values to discrete symbols. The residue between updated weights and the predicted version is inputted into the quantizer. Lastly, the implementation of entropy coding in [86] is arithmetic coding. This method encodes a tag representing the cumulative probability distribution of the input sequence. This tag lies within an interval determined by the cumulative probability distribution of the sequence. Throughout the coding process, a preset probability table guides the algorithm in adjusting and managing the tag, eventually yielding the tag as a binary bitstream. Table 15 presents the test loss on different coding schemes with communication cost = 1200 min without entropy coding on the non-i.i.d. Fashion-MNIST dataset and non-i.i.d. CIFAR-10 dataset. We can observe that the proposed scheme outperforms FedAvg [4], QSGD [107], STC [108], FedPAQ [109], and UVeQFed [110] in reducing communication loss.

Besides, run-length coding is used to encode the sparsified vector in [111], which is a simple form of data compression where consecutive identical data elements are replaced with a single data value and a count of the number of occurrences. In [108], [112], Golomb coding is employed to encode the distances between nonzero elements, a method used to encode non-negative integer values, particularly those with a geometric distribution. Elias omega coding is utilized in [113] to recursively encode the prefix of the binary representation.

Model compression algorithms often assume a fixed code length, which may not be ideal due to the heterogeneity and variability of model updates. In [87], Fed-CVLC (Federated Learning Compression with Variable-Length Codes) is proposed, which dynamically adjusts the code length based on the model update dynamics. The authors conduct extensive experiments with public datasets to evaluate Fed-CVLC's performance. The results demonstrate that Fed-CVLC significantly outperforms state-of-the-art methods, improving model utility by 1.50% to 5.44% or reducing communication traffic by 16.67% to 41.61%.

Considering that user data may not be independent, [103] introduces Content Compression Coding (CCC) to FL, which leverages the correlation between clients to compress data efficiently, aiming to reduce the transmission bandwidth of FL. Specifically, CCC divides clients into several groups, where clients are highly correlated. Each group has a head client, and the other clients in the group are encoded with reference to the head client. Content Compression Coding is applied to reduce the amount of transmitting information, and WinRAR, a common compression scheme, is adopted to evaluate the actual compression performance.

Over-the-air computation involves performing computational tasks directly on data as it is transmitted over a wireless communication channel, reducing the need for data transmission and enabling more efficient use of resources in wireless networks. A novel personalized FL framework based on deep coded over-the-air computation, named DipFL, is proposed in [88], consisting of a deep AirComp aggregation (DACA) module, a joint source-channel coding (JSCC) module and a personalized mix module. Specifically, DACA module is designed for n-to-1 data aggregation, JSCC module compresses the aggregated data to reduce the communication cost based on the variational auto-encoder (VAE) model

by encoding the transmitted data and introduces certain regularisation terms to reduce the bias of local samples, and the personalized mix module is used to alleviate the problem of non-iid data between clients by combining the global and local models at clients to initializes the local model before each iteration of local training.

In [114], they introduce a new coded over-the-air computation (codedAirComp) for model aggregation in FL over Gaussian multiple access channels (MAC). Specifically, the proposed codedAirComp compresses the local gradient by stochastic uniform quantization and utilizes nested lattice coding to transmit data, which improves FL performance compared with the traditional coding scheme.

Device-to-device (D2D) communication plays a crucial role in enabling efficient and scalable FL systems. A novel design of D2D-aided coded distributed learning method in [115], named D2D-CDL, is to reduce communication overhead and achieve efficient load balancing across devices in distributed learning by leveraging D2D communication to facilitate efficient data exchange among proximate devices. Data transmitted through D2D links undergoes three main operations: compression and coding, data exchange, and gradient computation. They derive optimal compression rates that minimize processing time and improve the efficiency of the distributed learning system considering system dynamics, diverse computational resources, and deployment characteristics. In addition, to maintain a certain level of data privacy, data transmitted on the D2D links will be compressed and coded by DP in advance.

C. PRIVACY AND SECURITY IN FL

In Section III-D, we have investigated various coding techniques aimed at enhancing privacy and security in DML, such as Lagrange coding, secure distributed matrix multiplication, coded state machines, and the introduction of artificial noise. In this section, we focus on those coding techniques that aim for privacy and security in FL. As we know, the initial purpose of FL is to protect the privacy of distributed clients when aggregating.

Among the traditional privacy-enhancing techniques are differential privacy (DP) [116] and secure multi-party computation (MPC) [117], while traditional security-enhancing techniques encompass the application of secure communication protocols, encryption techniques, and access controls. For example, the secure data aggregation protocol studied in [118] uses MPC protocol and can be applied to real-world FL scenarios. This protocol requires only one server, responsible for routing messages between other parties and computing the final result. It ensures only users' inputs are learned in aggregation and allows arbitrary users to leave at any time while maintaining security.

While federated learning is driven by the need for privacy protection, there are still many privacy problems in federated learning such as gradient leakage. Traditional technologies applying to privacy-preserving in FL are DP, homomorphic encryption (HE) [119] and secure MPC. Coding can also be used to protect privacy in the FL training process through techniques like encryption, which ensures the confidentiality of sensitive data by transforming it into an unreadable format.

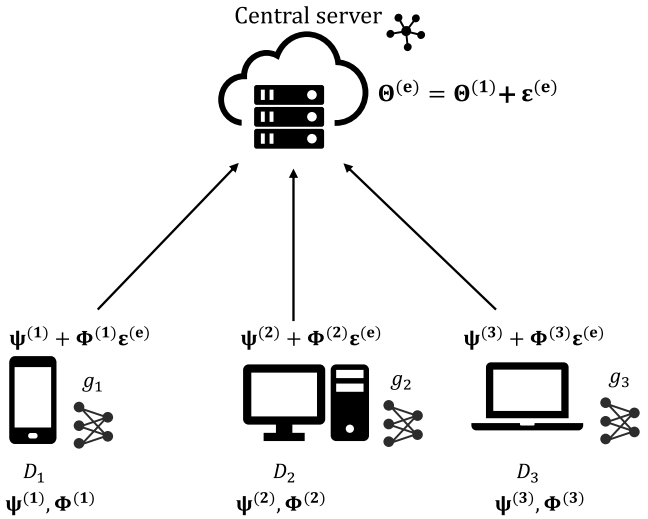


FIGURE 7. An example of CodedSecAgg with three devices based on (n, k) Shamir's Secret Sharing. Each device distributes n shares of the encoded data to the other $n-1$ devices. Device i have access to one share of the global data $\Psi^{(i)}$ and $\Phi^{(i)}$ and send out $\Psi^{(i)} + \Phi^{(i)}\epsilon^{(e)}$ to central server at epoch e. The central server can reconstruct the global model when receiving shares from at least k devices.

To enhance privacy in traditional FL and mitigate the impact of stragglers, [90] and [80] propose a novel secure aggregation scheme called CodedSecAgg. This scheme utilizes Shamir's secret sharing to introduce redundancy in device data. In (n, k) secret sharing, a secret is divided into n shares in a way that the secret can be reconstructed using any k of the shares, while fewer than k shares reveal no information about the secret. This concept is rooted in threshold cryptography, where the threshold k specifies the minimum number of shares required to reconstruct the secret.

CodedSecAgg comprises two phases as shown in Figure 7. In the first phase, devices employ (n, k) Shamir's Secret Sharing Scheme (SSS) to distribute their local data among other devices in the network, introducing data redundancy to handle stragglers. Based on CodedPaddedFL, device i encodes $\boldsymbol{G_i}^{(1)}$ together with its one-time pads into n shares $\{\boldsymbol{\Psi}_i^{(1)}, \ldots, \boldsymbol{\Psi}_i^{(n)}\}$ and $\mathbf{X}^{(i)\top}\mathbf{X}^{(i)}$ together with its one-time pads into n shares $\{\boldsymbol{\Phi}_i^{(1)}, \ldots, \boldsymbol{\Phi}_i^{(n)}\}$ using a nonsystematic (n, k) Reed-Solomon code. Then each device sends one share of each encoding to each of the other $n-1$ devices. When data sharing is done, device i computes $\boldsymbol{\Psi}^{(i)} = \sum_{j=1}^{n} \boldsymbol{\Psi}_j^{(i)}$ and $\boldsymbol{\Phi}^{(i)} = \sum_{j=1}^{n} \boldsymbol{\Phi}_j^{(i)}$. In the second phase, the devices exploit the linearity of Shamir's SSS by computing the gradient updates on $\tilde{\boldsymbol{G}}_i^{(e)} = \boldsymbol{\Psi}^{(i)} + \boldsymbol{\Phi}^{(i)}\boldsymbol{\epsilon}^{(e)}$ at epoch e. After receiving k devices, the central server can decode the global gradient base on SSS. During the learning process, the central server does not gain access to any local gradient, thereby preventing a model inversion attack.

The experiments conducted in [80] compare the performance of CodedSecAgg with LightSecAgg [120]

TABLE 16. Comparison of training time for different schemes when training accuracy reaches 95% with 100 devices [90].

Coding Schemes	Number of Colluding Agents z					
	1	5	10	20	30	60
CodedSecAgg [90]	2.23	2.68	2.71	2.88	3+	3+
LightSecAgg [120]	3+					
CodedPaddedFL [80]	1.67					

TABLE 17. Comparison of training time for different schemes when training accuracy reaches 95% with 1000 devices [90].

Coding Schemes	Number of Colluding Agents z					
	1	50	100	200	300	600
CodedSecAgg [90]	7.87	12.23	14.46	18.63	19.80	29.52
LightSecAgg [120]	30+					
CodedPaddedFL [80]	6.83					

under varying numbers of colluding agents z. From the results presented in Table 16 and Table 17, it is evident that CodedSecAgg is more sensitive to an increase in the security level z, whereas LightSecAgg remains largely unaffected. In all settings of z, CodedSecAgg outperforms LightSecAgg.

However, there is a trade-off between privacy and communication costs in CodedSecAgg. Specifically, in Table 16, it is shown that for achieving an accuracy of 95% on the MNIST dataset and preventing a model inversion attack, CodedSecAgg incurs a moderate additional latency of 34% compared to CodedPaddedFL. On the other hand, as seen in Table 17, when considering 1000 devices collaboratively training, CodedSecAgg with a privacy level of $z = 1$ requires only 15% more latency compared to CodedPaddedFL. This implies that the higher cost of the sharing phase in CodedSecAgg compared to CodedPaddedFL becomes negligible in the long run.

Due to the privacy leakage of client information, [91] develops a new approach named Lagrange Coding with Mask (LCM) for secure aggregation across multiple servers. This approach operates within a threat model based on Shannon's information-theoretic security framework, which prevents colluding servers from deducing any details about the client's data or its aggregated value. Likewise, colluding clients are unable to access other honest clients' data. To address these threats, LCM employs client-side masking to reduce the risk of exposing client information to the server.

A new FL model based on Lagrange coded computing is proposed in [92]. This model enhances security against system noise, such as wireless channel interference and malicious vehicles, by introducing computational redundancy to the data processed by vehicles in an FL-enabled Internet of Vehicles (IoV) system. While existing LCC models with Reed-Solomon decoders are suitable for tasks involving polynomial functions, the paper proposes a low-complexity polynomial approximation of the machine learning results, which are typically non-polynomial functions. The simulation results for traffic slowness prediction demonstrate that the proposed LCoFL model not only achieves high accuracy with the proposed approximation method but also effectively addresses the erroneous machine learning results produced by malicious vehicles.

In contrast to the work in [93], [121] presents a coded FL approach that is resilient to stragglers. This scheme not only preserves the privacy level of conventional FL but also ensures computational privacy. The proposed method introduces redundancy by enabling D2D communication, allowing slower devices to share their data with faster ones without actively participating in the learning process. To maintain privacy, the scheme employs the code-based McEliece cryptosystem during the data sharing phase. Subsequently, faster devices compute local gradients on the encrypted data, and the central server aggregates these encrypted gradients to update the model.

The utilization of coding techniques represents a revolutionary approach to streamlining communication and tackling inherent obstacles. Specifically, encoding methods, including erasure-correcting codes, are strategically integrated into the federated learning framework to optimize the transmission of model updates between the central server and decentralized edge devices. The adoption of coding techniques in FL is a pivotal innovation, fostering improved communication efficiency, privacy preservation, and resilience against challenges such as straggling devices. Through the systematic integration of coding methodologies, FL systems benefit from enhanced reliability, security, and performance, positioning coding as a crucial element in advancing collaborative model training in decentralized environments.

V. EVOLUTION OF CODES AND REAL-WORLD APPLICATIONS

In this section, we begin by delineating the evolution of coding techniques, followed by an exploration of their real-world applications in technologies such as MEC or IoT devices [122], where coding techniques are leveraged in DML to tackle various challenges.

A. EVOLUTION OF CODES

Now, we summarize the evolution of coding techniques. The concept of coded computing is introduced by Li and Avestimehr [123] to mitigate fundamental bottlenecks in large-scale distributed computing and machine learning, including issues related to communication bandwidth, stragglers and privacy or security. CDC is implemented in the context of MapReduce-based architectures, showcasing its potential to enhance efficiency in distributed learning frameworks.

1) CLASSIC CODES

ECCs are essential components of communication systems, providing mechanisms to detect and correct errors that

may arise during data transmission. By incorporating redundancy into transmitted data, ECC ensures the integrity and reliability of communication channels, thereby enabling error-free transmission in the presence of noise and channel impairments.

The repetition scheme represents one of the earliest techniques employed in communication systems. It involves the duplication of data bits to enhance reliability in noisy channels. This approach encompasses two variations used in gradient coding: fractional repetition, where all groups are identical replicas of each other, and cyclic repetition, which cyclically assigns partitions to each node. Despite its simplicity and straightforward implementation, the repetition scheme suffers from inefficiency in terms of bandwidth utilization and error correction capabilities.

MDS codes are foundational in error correction, providing optimal resilience against channel impairments and data corruption. Distinguished by their maximum distance properties, these codes facilitate effective error detection and correction within communication systems. Taking advantage of the "any K of N" characteristic of MDS codes, computations can proceed as soon as any K of the fastest nodes complete their tasks. This feature allows the system to cope with up to $N - K$ arbitrary stragglers, enhancing the robustness.

Utilizing Lagrange interpolation polynomial, LCC constructs computational redundancy in a distinctive coded format distributed among the workers. Its universality stems from the encoding process being agnostic to the output function. Moreover, when fresh data emerges and updates are imperative for coded data batches, solely the new data undergoes encoding and is added to the pre-existing coded batches. This avoids accessing the whole uncoded dataset and re-encoding it for updating the coded data.

Take gradient coding and its advances as an example, gradient coding [7] is a solution to address the challenges posed by stragglers in distributed learning systems. By introducing redundancy in coded local partial derivatives, gradient coding effectively mitigates the impact of stragglers, enhancing the efficiency of distributed learning algorithms. The fractional repetition scheme and cyclic repetition scheme are initially employed, leveraging the MDS property within the coding framework.

Later, researchers developed improved Gradient coding techniques to optimize distributed learning processes. These approaches incorporate advanced algorithms and optimizations to enhance fault tolerance and accelerate convergence in distributed environments. For example, in some advanced GC algorithms, to maximize the computational potential, the incomplete computational results of partial stragglers are leveraged instead of being dropped.

2) MODERN CODES

With the evolution of modern coding theory, an increasing array of codes is being implemented in the transmission of DML. Recent advancements in ECCs have led to the development of advanced coding techniques, such as LDPC codes, Polar codes, and Fountain codes. These modern coding schemes offer improved error correction capabilities, increased efficiency, and enhanced performance in challenging communication environments. For example, Fountain codes, a notable example in modern coding theory, offer efficient and robust data transmission capabilities, particularly in scenarios with varying channel conditions and packet losses. Hybrid coding schemes have also been proposed to strike a balance between these advantages.

The evolution of coding techniques in communication engineering has been marked by continuous innovation and advancements. From classical methods like repetition schemes to modern approaches such as Fountain codes, coding techniques have played an important role in shaping the performance and reliability of communication systems.

B. REAL-WORLD APPLICATIONS

Smartphones, smart vehicular nodes (including cars and roadside units (RSUs)), wearable, laptops, and unmanned aerial vehicles (UAVs) can work as edge devices in real-world MEC systems, leveraging their significant storage, communication, and computational capacities. However, latency and bandwidth limitations necessitate the processing of large amounts of data locally in a distributed manner. This approach finds relevance in various applications such as healthcare, intelligent transportation, and industrial automation. Taking medical monitoring equipment as an example [124], the device comprehensively analyzes and learns the collected monitoring data of the individual being monitored, such as heartbeat, blood oxygen, blood pressure, and breath to predict health events. It also learns posture data to detect whether a fall has occurred and takes emergency measures as well as calls for assistance. In scenarios where data privacy is paramount, the low-latency FL scheme [79] offers a feasible solution, particularly in wireless edge networks, through the application of coded computing techniques.

Coding suits for communication scenarios with poor link quality leading to high error rates. The Doppler effect caused by high-speed movement, as well as the occlusion between vehicles, results in unstable communication links in vehicle-to-vehicle (V2V) networks. The introduction of coding schemes can effectively enhance robustness and reduce the retransmission in V2V communication [47].

The heterogeneous nature of mobile IoT networks presents significant differences in hardware, computational capability, and power constraints. While recent works have implemented coded computation on edge devices, evaluations have largely been limited to simulations that do not fully capture real-world challenges such as communication overhead and battery consumption. To address the challenges posed by limited computational resources, battery life, and data privacy, [70] proposes lightweight secure coded distributed learning algorithms optimized for high energy efficiency and low computational complexity in mobile IoT using real-world commercial hardware with the development of a

prototype for Android platforms that can be further deployed. Through extensive experiments on multiple commercial mobile IoT networks, they demonstrate that the algorithms protect data privacy while significantly reducing both runtime and battery consumption.

In the Internet of Mobile Things, mobile devices such as mobile phones, robots, and smart vehicles have predictable or unpredictable movement positions, [48] considered mobility and used fountain code to obtain lower total processing time and latency.

Although coding is a promising technique and proved efficient theoretically, it still faces some limitations in practical applications. Now, we discuss the practical challenges of implementing these coding techniques:

- Energy. Encoding and decoding processes are energy consuming while some edge nodes are energy sensitive, urging for power management protocol [125].
- Computation resources. Limited computation resources and the large computational workload of training ML models. Some work nodes like RSUs in vehicle-to-vehicle networks have sufficient resources to train specific ML models, but some nodes such as sensors do not. So, load allocation needs to be considered.
- Balancing computation latency and communication latency. Computation latency and communication latency have a trade-off, parameters need to be designed to balance it to obtain the minimum overall latency.
- Heterogeneity of the MEC systems. Edge nodes in a MEC system may have heterogeneous data, leading to the protocol for heterogeneity [13].

Future research in DML and FL could focus on addressing the challenges posed by untrustworthy networks, such as data collusion or wiretapping, and explore methods to preserve data privacy using coding techniques. Additionally, given the constraints of computation and energy resources, there is a critical need for energy-efficient algorithms. In the context of MEC and mobile IoT, where network devices are dynamic and heterogeneous, algorithms must adapt to fluctuations in the number of connected worker devices with varying computational capabilities, ensuring scalability.

VI. CONCLUSION

In conclusion, this survey focused on the development of applying coding techniques in distributed systems, including distributed coded machine learning and coding in federated learning. These paradigms distribute tasks among multiple workers, alleviating the resource burden on individual nodes. However, the inherent challenges of slow nodes and failed links, known as stragglers, for better system performance, lower overall runtime, enhanced robustness or privacy. The incorporation of redundancy through coding presents a compelling tradeoff between computation and communication costs. Notably, coding emerges as a promising technique for constructing distributed systems that are both robust and secure, achieving low-latency performance. As we navigate the landscape of distributed learning, coding stands out as a valuable tool for optimizing the efficiency and reliability of these systems. Future works may include a better coding scheme or task allocation method. Theoretically, the trade-off between computation and communication can be further studied.

REFERENCES

[1] K. T. Kim, C. Joe-Wong, and M. Chiang, "Coded edge computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2020, pp. 237–246.

[2] (Eur. Union, Brussels, Belgium). *General Data Protection Regulation*. (2018). [Online]. Available: https://gdpr-info.eu/

[3] "Cybersecurity law of the people's republic of China: National people's congress." 2021. [Online]. Available: https://www.newamerica.org/cybersecurity-initiative/digichina/blog/translation-cybersecurity-law-peoples-republic-china/

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[5] S. Dhakal, S. Prakash, Y. Yona, S. Talwar, and N. Himayat, "Coded federated learning," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2019, pp. 1–6.

[6] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[7] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 3368–3376.

[8] S. Ji, Z. Zhang, Z. Yang, R. Jin, and Q. Yang, "Coded parallelism for distributed deep learning," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2023, pp. 2410–2415.

[9] H. Chen, Y. Ye, M. Xiao, M. Skoglund, and H. V. Poor, "Coded stochastic ADMM for Decentralized consensus optimization with edge computing," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5360–5373, Apr. 2021.

[10] T. Chen, G. B. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5055–5065.

[11] J. Zhang and O. Simeone, "LAGC: Lazily aggregated gradient coding for straggler-tolerant and communication-efficient distributed learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 3, pp. 962–974, Mar. 2021.

[12] R. Bitar, M. Wootters, and S. E. Rouayheb, "Stochastic gradient coding for flexible straggler mitigation in distributed learning," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2019, pp. 1–5.

[13] S. Dhakal, S. Prakash, Y. Yona, S. Talwar, and N. Himayat, "Coded computing for distributed machine learning in wireless edge network," in *Proc. IEEE 90th Veh. Technol. Conf. (VTC)*, 2019, pp. 1–6.

[14] D.-J. Han, J.-Y. Sohn, and J. Moon, "Hierarchical broadcast coding: Expediting distributed learning at the wireless edge," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2266–2281, Apr. 2021.

[15] S. Prakash, A. Reisizadeh, R. Pedarsani, and A. S. Avestimehr, "Hierarchical coded gradient aggregation for learning at the edge," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2020, pp. 2616–2621.

[16] H. Wang et al., "Heterogeneity-aware gradient coding for tolerating and leveraging stragglers," *IEEE Trans. Comput.*, vol. 71, no. 4, pp. 779–794, Apr. 2022.

[17] E. Ozfatura, D. Gündüz, and S. Ulukus, "Speeding up distributed gradient descent by Utilizing non-persistent stragglers," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2019, pp. 2729–2733.

[18] E. Ozfatura, D. Gündüz, and S. Ulukus, "Gradient coding with clustering and multi-message communication," in *Proc. IEEE Data Sci. Workshop (DSW)*, 2019, pp. 42–46.

[19] B. Buyukates, E. Ozfatura, S. Ulukus, and D. Gündüz, "Gradient coding with dynamic clustering for straggler-tolerant distributed learning," *IEEE Trans. Commun.*, vol. 71, no. 6, pp. 3317–3332, Jun. 2023.

[20] L. Tauz and L. Dolecek, "Multi-message gradient coding for utilizing non-persistent stragglers," in *Proc. 53rd Asilomar Conf. Signals, Syst., Comput.*, 2019, pp. 2154–2159.

[21] S. Kadhe, O. O. Koyluoglu, and K. Ramchandran, "Communication-efficient gradient coding for straggler mitigation in distributed learning," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2020, pp. 2634–2639.

[22] R. K. Maity, A. S. Rawa, and A. Mazumdar, "Robust gradient descent via moment encoding and LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2019, pp. 2734–2738.

[23] N. S. Ferdinand and S. C. Draper, "Anytime coding for distributed computation," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, 2016, pp. 954–960.

[24] X. Su, B. Sukhnandan, and J. Li, "On arbitrary ignorance of stragglers with gradient coding," in *Proc. IEEE 43rd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2023, pp. 660–670.

[25] Q. Wang, Y. Cui, C. Li, J. Zou, and H. Xiong, "Optimization-based block coordinate gradient coding for mitigating partial stragglers in distributed learning," *IEEE Trans. Signal Process.*, vol. 71, pp. 1023–1038, Feb. 2023.

[26] J. Zhu, Y. Pu, V. Gupta, C. Tomlin, and K. Ramchandran, "A sequential approximation framework for coded distributed optimization," in *Proc. 55th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, 2017, pp. 1240–1247.

[27] M. N. Krishnan, E. Hosseini, and A. Khisti, "Sequential gradient coding for packet-loss networks," *IEEE J. Select. Areas Inf. Theory*, vol. 2, no. 3, pp. 919–930, Sep. 2021.

[28] X. Wang et al., "Two-stage coded distributed learning: A dynamic partial gradient coding perspective," in *Proc. IEEE 43rd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2023, pp. 942–952.

[29] X. Xu, X. Lin, and L. Duan, "Design and performance analysis of partial computation output schemes for accelerating coded machine learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 2, pp. 1119–1130, Mar./Apr. 2023.

[30] C. Karakus, Y. Sun, and S. Diggavi, "Encoded distributed optimization," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 2890–2894.

[31] V. Gupta, S. Kadhe, T. Courtade, M. W. Mahoney, and K. Ramchandran, "OverSketched Newton: Fast convex optimization for serverless systems," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2020, pp. 288–297.

[32] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4227–4242, Jul. 2019.

[33] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Communication-aware computing for edge processing," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 2885–2889.

[34] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Straggler mitigation in distributed optimization through data encoding," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5440–5448.

[35] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Redundancy techniques for straggler mitigation in distributed optimization and learning," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 2619–2665, Jan. 2019.

[36] Y. Yang, P. Grover, and S. Kar, "Coded distributed computing for inverse problems," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 709–719.

[37] S. Dutta, Z. Bai, H. Jeong, T. M. Low, and P. Grover, "A unified coded deep neural network training strategy based on generalized PolyDot codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1585–1589.

[38] L. Liu, A. Solomon, S. Salamatian, and M. Médard, "Neural network coding," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6.

[39] N. Zhang and M. Tao, "An adaptive distributed source coding design for distributed learning," in *Proc. 13th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2021, pp. 1–5.

[40] B. Bartan and M. Pilanci, "Randomized polar codes for anytime distributed machine learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 4, pp. 393–404, Sep. 2023.

[41] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, 1960. [Online]. Available: https://doi.org/10.1137/0108018

[42] R. Bitar, M. Wootters, and S. E. Rouayheb, "Stochastic gradient coding for straggler mitigation in distributed learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 277–291, May 2020.

[43] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4427–4437.

[44] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 1215–1225. [Online]. Available: https://proceedings.mlr.press/v89/yu19b.html

[45] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 34–40, Apr. 2017.

[46] J. Zhang and O. Simeone, "On model coding for distributed inference and transmission in mobile edge computing systems," *IEEE Commun. Lett.*, vol. 23, no. 6, pp. 1065–1068, Jun. 2019.

[47] F. Chen, Z. Gao, Z. Liu, L. Huang, and Y. Tang, "Code-based computation offloading in vehicular fog networks," in *Proc. Int. Conf. Commun., Comput., Cybersecur., Inform. (CCCI)*, 2021, pp. 1–5.

[48] J. Yue and M. Xiao, "Coding for distributed fog computing in Internet of Mobile Things," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1337–1350, Apr. 2021.

[49] K. Liang and Y. Wu, "Two-layer coded gradient aggregation with straggling communication links," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2021, pp. 1–5.

[50] H. Esfahanizadeh, A. Cohen, and M. Medard, "Stream iterative distributed coded computing for learning applications in heterogeneous systems," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2022, pp. 230–239.

[51] T. Li, J. Zhang, and X. He, "Coding-aware rate splitting for distributed coded edge learning," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2023, pp. 1–6.

[52] N. Ding, Z. Fang, L. Duan, and J. Huang, "Optimal incentive and load design for distributed coded machine learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2090–2104, Jul. 2021.

[53] H. Hu, Y. Wu, Y. Shi, S. Li, C. Jiang, and W. Zhang, "Communication-efficient coded computing for distributed multi-task learning," *IEEE Trans. Commun.*, vol. 71, no. 7, pp. 3861–3875, Jul. 2023.

[54] Y. Zhou, Q. Ye, and H. Huang, "DRL-based workload allocation for distributed coded machine learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2023, pp. 3175–3180.

[55] Y. Yang, P. Grover, and S. Kar, "Coding for a single sparse inverse problem," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2018, pp. 1575–1579.

[56] Y. Yang, M. Chaudhari, P. Grover, and S. Kar, "Coded iterative computing using substitute decoding," 2018, *arXiv:1805.06046*.

[57] S. Dutta, V. Cadambe, and P. Grover, "'Short-dot': Computing large linear transforms distributedly using coded short dot products," *IEEE Trans. Inf. Theory*, vol. 65, no. 10, pp. 6171–6193, Oct. 2019.

[58] A. B. Das, A. Ramamoorthy, and N. Vaswani, "Efficient and robust distributed matrix computations via convolutional coding," *IEEE Trans. Inf. Theory*, vol. 67, no. 9, pp. 6266–6282, Sep. 2021.

[59] X. Lou, H. Yao, C. W. Tan, and J. Wang, "VANDER: Efficient cooperative watchdog monitoring for lossy wireless network coding," *IEEE Trans. Veh. Technol.*, vol. 64, no. 2, pp. 702–713, Feb. 2015.

[60] A. K. Pradhan, A. Heidarzadeh, and K. R. Narayanan, "Factored LT and factored raptor codes for large-scale distributed matrix multiplication," *IEEE J. Select. Areas Inf. Theory*, vol. 2, no. 3, pp. 893–906, Sep. 2021.

[61] J. Kosaian, K. V. Rashmi, and S. Venkataraman, "Parity models: Erasure-coded resilience for prediction serving systems," in *Proc. 27th ACM Symp. Oper. Syst. Princ.*, 2019, pp. 30–46. [Online]. Available: https://doi.org/10.1145/3341301.3359654

[62] J. Kosaian, K. V. Rashmi, and S. Venkataraman, "Learning-based coded computation," *IEEE J. Select. Areas Inf. Theory*, vol. 1, no. 1, pp. 227–236, May 2020.

[63] N. Agrawal et al., "A learning-based approach to approximate coded computation," in *Proc. IEEE Inf. Theory Workshop (ITW)*, 2022, pp. 600–605.

[64] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Coded computing in unknown environment via online learning," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2020, pp. 185–190.

[65] S. Yang, A. Hareedy, R. Calderbank, and L. Dolecek, "Hierarchical hybrid error correction for time-sensitive devices at the edge," 2021, *arXiv:2103.11046*.

[66] S. Yang, A. Hareedy, R. Calderbank, and L. Dolecek, "Hierarchical coding to enable scalability and flexibility in heterogeneous cloud storage," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[67] S. Yang, A. Hareedy, R. Calderbank, and L. Dolecek, "Hierarchical coding for cloud storage: Topology-adaptivity, scalability, and flexibility," *IEEE Trans. Inf. Theory*, vol. 68, no. 6, pp. 3657–3680, Jun. 2022.

[68] L. Tauz and L. Dolecek, "Variable coded batch matrix multiplication," *IEEE J. Select. Areas Inf. Theory*, vol. 3, no. 2, pp. 306–320, Jun. 2022.

[69] J. So, B. Guler, and A. S. Avestimehr, "CodedPrivateML: A fast and privacy-preserving framework for distributed machine learning," *IEEE J. Select. Areas Inf. Theory*, vol. 2, no. 1, pp. 441–451, Mar. 2021.

[70] Y. Yang, R. G. D'Oliveira, S. E. Rouayheb, X. Yang, H. Seferoglu, and Y. Chen, "Secure coded computation for efficient distributed learning in mobile IoT," in *Proc. 18th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, 2021, pp. 1–9.

[71] S. Li, S. Sahraei, M. Yu, S. Avestimehr, S. Kannan, and P. Viswanath, "Coded state machine—Scaling state machine execution under Byzantine faults," in *Proc. ACM Symp. Princ. of Distrib. Comput.*, 2019, pp. 150–152.

[72] Y. Xue, X. Lin, J. Wu, and J. Li, "Wireless coded distributed learning with gaussian-based local differential privacy," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2023, pp. 1943–1948.

[73] T. Tang, R. E. Ali, H. Hashemi, T. Gangwani, S. Avestimehr, and M. Annavaram, "Adaptive verifiable coded computing: Towards fast, secure and private distributed machine learning," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2022, pp. 628–638.

[74] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 56–67, Oct. 1998. [Online]. Available: https://doi.org/10.1145/285243.285258

[75] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008. [Online]. Available: https://doi.org/10.1145/1327452.1327492

[76] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2019, *arXiv:1810.04805*.

[77] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[78] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends® Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[79] S. Prakash et al., "Coded computing for low-latency federated learning over wireless edge networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 233–250, Jan. 2021.

[80] R. Schlegel, S. Kumar, E. Rosnes, and A. G. I. Amat, "CodedPaddedFL and CodedSecAgg: Straggler mitigation and secure aggregation in federated learning," *IEEE Trans. Commun.*, vol. 71, no. 4, pp. 2013–2027, Apr. 2023.

[81] Y. Sun, J. Shao, Y. Mao, S. Li, and J. Zhang, "Stochastic coded federated learning: Theoretical analysis and incentive mechanism design," *IEEE Trans. Wireless Commun.*, vol. 23, no. 6, pp. 6623–6638, Jun. 2024.

[82] W. J. Yun et al., "SlimFL: Federated learning with superposition coding over slimmable neural networks," *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 2499–2514, Dec. 2023.

[83] B. Sasidharan and A. Thomas, "Coded gradient aggregation: A tradeoff between communication costs at edge nodes and at helper nodes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2021, pp. 2286–2291.

[84] B. Sasidharan and A. Thomas, "Coded gradient aggregation: A tradeoff between communication costs at edge nodes and at helper nodes," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 3, pp. 761–772, Mar. 2022.

[85] M. N. Krishnan, A. Thomas, and B. Sasidharan, "Hierarchical coded gradient aggregation based on layered MDS codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2023, pp. 2547–2552.

[86] K. Yue, R. Jin, C.-W. Wong, and H. Dai, "Communication-efficient federated learning via predictive coding," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 3, pp. 369–380, Apr. 2022.

[87] X. Su, Y. Zhou, L. Cui, J. Lui, and J. Liu, "Fed-CVLC: Compressing federated learning communications with variable-length codes," 2024, *arXiv:2402.03770*.

[88] D. Chen, M. Lei, M.-M. Zhao, A. Liu, and S. Sheng, "Deep learning based coded over-the-air computation for personalized federated learning," in *Proc. IEEE 98th Veh. Technol. Conf. (VTC)*, 2023, pp. 1–5.

[89] L. Kong, H. Tao, J. Wang, Z. Huang, and J. Xiao, "Network coding for federated learning systems," in *Proc. 27th Int. Conf., Neural Inf. Process. (ICONIP)*, 2020, pp. 546–557.

[90] R. Schlegel, S. Kumar, E. Rosnes, and A. G. I. Amat, "Straggler-resilient secure aggregation for federated learning," in *Proc. 30th Eur. Signal Process. Conf. (EUSIPCO)*, 2022, pp. 712–716.

[91] K. Liang, S. Li, M. Ding, and Y. Wu, "Multi-server secure aggregation with unreliable communication links," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2023, pp. 2560–2565.

[92] W. Ni et al., "Lagrange coded federated learning (L-CoFL) model for internet of vehicles," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2022, pp. 864–872.

[93] M. Xhemrishi, A. G. I. Amat, E. Rosnes, and A. Wachter-Zeh, "Computational code-based privacy in coded federated learning," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2022, pp. 2034–2039.

[94] Y. Sun, J. Shao, S. Li, Y. Mao, and J. Zhang, "Stochastic coded federated learning with convergence and privacy guarantees," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2022, pp. 2028–2033.

[95] A. Anand, S. Dhakal, M. Akdeniz, B. Edwards, and N. Himayat, "Differentially private coded federated linear regression," in *Proc. IEEE Data Sci. Learn. Workshop (DSLW)*, 2021, pp. 1–6.

[96] A. Motamedi, S. Yun, J.-M. Kang, Y. Ge, and I.-M. Kim, "Redundancy management in federated learning for fast communication," *IEEE Trans. Commun.*, vol. 71, no. 11, pp. 6332–6347, Nov. 2023.

[97] M. Dai, Z. Zheng, Z. Hong, S. Zhang, and H. Wang, "Edge computing-aided coded vertical federated linear regression," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 3, pp. 1543–1551, Sep. 2022.

[98] Y. Yang, J. Wang, and F. Gu, "Linear coded federated learning under multiple stragglers over heterogeneous clients," in *Proc. IEEE 25th Int. Conf. Comput. Support. Cooperative Work Design (CSCWD)*, 2022, pp. 1221–1226.

[99] D. Wang et al., "CFL-HC: A coded federated learning framework for heterogeneous computing scenarios," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2021, pp. 1–6.

[100] W. Halbawi, Z. Liu, and B. Hassibi, "Balanced reed-solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2016, pp. 935–939.

[101] N. Charalambides, M. Pilanci, and A. O. Hero, "Securely aggregated coded matrix inversion," *IEEE J. Select. Areas Inf. Theory*, vol. 4, pp. 405–419, Sep. 2023.

[102] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Proc. 6th IEEE Int. Symp. Netw. Comput. Appl. (NCA)*, 2007, pp. 79–86.

[103] K. Deng, Z. Chen, S. Zhang, C. Gong, and J. Zhu, "Content compression coding for federated learning," in *Proc. 11th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2019, pp. 1–6.

[104] H. Hu, S. Li, M. Cheng, S. Ma, Y. Shi, and Y. Wu, "On exploiting network topology for hierarchical coded multi-task learning," *IEEE Trans. Commun.*, early access, Mar. 25, 2024, doi: 10.1109/TCOMM.2024.3381671.

[105] Y. Shi, Z. Zhu, P. Fan, K. B. Letaief, and C. Peng, "FedNC: A secure and efficient federated learning method inspired by network coding," 2024, *arXiv:2305.03292*.

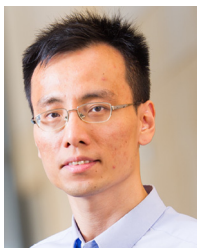
[106] K. Sayood, *Introduction to Data Compression*. Cambridge, MA, USAL: Morgan Kaufmann, 2017.

[107] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. 31st Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–12.
[108] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.
[109] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. Int. Conf. Artif. Intell. Statist.*, Jan. 2020, pp. 2021–2031.
[110] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "UVeQFed: Universal vector quantization for federated learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 500–514, 2021.
[111] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.
[112] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2019, pp. 1–8.
[113] Z. Zhu, Y. Shi, G. Xin, C. Peng, P. Fan, and K. B. Letaief, "Towards efficient federated learning: Layer-wise pruning-Quantization scheme and coding design," *Entropy*, vol. 25, no. 8, p. 1205, 2023. [Online]. Available: https://www.mdpi.com/1099-4300/25/8/1205
[114] N. Zhang, M. Tao, J. Wang, and S. Shao, "Coded over-the-air computation for model aggregation in federated learning," *IEEE Commun. Lett.*, vol. 27, no. 1, pp. 160–164, Jan. 2023.
[115] N. Zeulin, O. Galinina, N. Himayat, S. Andreev, and R. W. Heath, "Dynamic network-assisted D2D-aided coded distributed learning," *IEEE Trans. Commun.*, vol. 71, no. 6, pp. 3352–3367, Jun. 2023.
[116] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends® Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2014.
[117] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci. (SFCS)*, 1982, pp. 160–164.
[118] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191. [Online]. Available: https://doi.org/10.1145/3133956.3133982
[119] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.
[120] J. So et al., "Lightsecagg: A lightweight and versatile design for secure aggregation in federated learning," in *Proc. Mach. Learn. Syst.*, 2022, pp. 694–720.
[121] S. Kumar, R. Schlegel, E. Rosnes, and A. G. I. Amat, "Coding for straggler mitigation in federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2022, pp. 4962–4967.
[122] B. Sudharsan et al., "Toward distributed, global, deep learning using IoT devices," *IEEE Internet Comput.*, vol. 25, no. 3, pp. 6–12, May/Jun. 2021.
[123] S. Li and S. Avestimehr, "Coded computing: Mitigating fundamental bottlenecks in large-scale distributed computing and machine learning," *Found. Trends® Commun. Inf. Theory*, vol. 17, no. 1, pp. 1–148, 2020. [Online]. Available: http://dx.doi.org/10.1561/0100000103
[124] S. Aminizadeh et al., "The applications of machine learning techniques in medical data processing based on distributed computing and the Internet of Things," *Comput. Methods Programs Biomed.*, vol. 241, Nov. 2023, Art. no. 107745. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016926072300411X
[125] M. T. Beck, S. Feld, A. Fichtner, C. Linnhoff-Popien, and T. Schimper, "ME-VoLTE: Network functions for energy-efficient video transcoding at the mobile edge," in *Proc. 18th Int. Conf. Intell. Next Gener. Netw.*, 2015, pp. 38–44.

YIQIAN ZHANG received the B.E. degree in communication engineering from the School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen, China, in 2021, where she is currently pursuing the Ph.D. degree in information and communication engineering under the supervision of Prof. Congduan Li. Her recent research interests include information theory and network coding.

TIANLI GAO received the B.E. degree from the School of Electronics and Communication Engineering, Sun Yat-sen University, Shenzhen, China, in 2022, where she is currently pursuing the master's degree in electronic and information engineering. Her research interests include information theory, network coding, and coded computation.

CONGDUAN LI (Senior Member, IEEE) received the B.S. degree in electrical engineering from the University of Science and Technology Beijing, China, in 2008, the M.S. degree in electrical engineering from Northern Arizona University, AZ, USA, in 2011, and the Ph.D. degree in electrical engineering from Drexel University, PA, USA, in 2015. From October 2015 to August 2018, he was a Postdoctoral Research Fellow with the Institute of Network Coding, The Chinese University of Hong Kong, and the Department of Computer Science, City University of Hong Kong. He is currently an Associate Professor with the School of Electronics and Communication Engineering, Sun Yat-sen University, China. His research interests lie in the broad areas related with networks, such as coding, security, wireless, storage, and caching.



CHEE WEI TAN (Senior Member, IEEE) received the M.A. and Ph.D. degrees in electrical engineering from Princeton University. He is currently an Associate Professor with Nanyang Technological University, Singapore. His research interests are in networks, distributed optimization, and generative AI. He is serving or has served as an Editor for the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, IEEE/ACM TRANSACTIONS ON NETWORKING, and IEEE TRANSACTIONS ON COMMUNICATIONS and a Distinguished Lecturer for IEEE COMMUNICATIONS SOCIETY.