Received 8 December 2022, accepted 29 December 2022, date of publication 5 January 2023, date of current version 11 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3234761

SURVEY

Enabling All In-Edge Deep Learning: A Literature Review

PRAVEEN JOSHI[1], MOHAMMED HASANUZZAMAN[1], CHANDRA THAPA[2], (Member, IEEE), HAITHEM AFLI[1], AND TED SCULLY[1]

[1]Computer Science Department, Munster Technological University, Cork, T12 P928 Ireland
[2]CSIRO Data61, Marsfield, NSW 2122, Australia

Corresponding author: Praveen Joshi (praveen.joshi@mycit.ie)

This research was conducted with the financial support of the ADVANCE CRT Ph.D. Cohort under Grant Agreement No.18/CRT/6222 and the ADAPT SFI Research Centre at Munster Technological University. The ADAPT SFI Centre for Digital Media Technology is funded by Science Foundation Ireland through the SFI Research Centres Programme and is co-funded under the European Regional Development Fund (ERDF) through Grant 13/RC/2106_P2. The project was partially supported by the Horizon 2020 projects STOP Obesity Platform under Grant Agreement No. 823978 and ITFLOWS under Grant Agreement No. 882986.

ABSTRACT In recent years, deep learning (DL) models have demonstrated remarkable achievements on non-trivial tasks such as speech recognition, image processing, and natural language understanding. One of the significant contributors to the success of DL is the proliferation of end devices that act as a catalyst to provide data for data-hungry DL models. However, computing DL training and inference still remains the biggest challenge. Moreover, most of the time central cloud servers are used for such computation, thus opening up other significant challenges, such as high latency, increased communication costs, and privacy concerns. To mitigate these drawbacks, considerable efforts have been made to push the processing of DL models to edge servers (a mesh of computing devices near end devices). Recently, the confluence point of DL and edge has given rise to edge intelligence (EI), defined by the International Electrotechnical Commission (IEC) as the concept where the data is acquired, stored, and processed utilizing edge computing with DL and advanced networking capabilities. Broadly, EI has six levels of categories based on the three locations where the training and inference of DL take place, *e.g.*, cloud server, edge server, and end devices. This survey paper focuses primarily on the fifth level of EI, called *all in-edge* level, where DL training and inference (deployment) are performed solely by edge servers. All in-edge is suitable when the end devices have low computing resources, *e.g.*, Internet-of-Things, and other requirements such as latency and communication cost are important such as in mission-critical applications (*e.g.*, health care). Besides, 5G/6G networks are envisioned to use all in-edge. Firstly, this paper presents all in-edge computing architectures, including centralized, decentralized, and distributed. Secondly, this paper presents enabling technologies, such as model parallelism, data parallelism, and split learning, which facilitates DL training and deployment at edge servers. Thirdly, model adaptation techniques based on model compression and conditional computation are described because the standard cloud-based DL deployment cannot be directly applied to all in-edge due to its limited computational resources. Fourthly, this paper discusses eleven key performance metrics to evaluate the performance of DL at all in-edge efficiently. Finally, several open research challenges in the area of all in-edge are presented.

INDEX TERMS Artificial intelligence, all in-edge, deep learning, distributed systems, decentralized systems, edge intelligence.

I. INTRODUCTION

The global community is increasingly becoming a data-driven environment in which end devices are generating vast quantities of data outside of the traditional data centers. International Telecommunication Union anticipates that global internet traffic per month will reach 607 Exabytes (EB) in 2025 and 5016 EB in 2030 [1]. This enormous amount of data has a positive impact on artificial intelligence (AI) applications. In particular, deep learning (DL) relies on the

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Da Lin.

availability of large quantities of data for its development, including training and inference [2], [3].

DL has shown promising progress in natural language processing, computer vision, and big data analysis in recent years. For example, DL models, such as BERT, Megatron-LM, GPT-3, and Gropher, are reaching a human-level understanding of the textual data in natural language processing tasks [4]. Moreover, DL models have exceeded human performance on various tasks, including object classification tasks [5], [6] and real-time strategy games [7].

DL training and deployment in the majority of scenarios use a centralized cloud-based structure. However, the need to collect, process, and transfer vast data to the central cloud often becomes a bottleneck in many mission-critical use cases [8], [9]. In this regard, edge computing provides a high-performance bridge from local systems to private and public clouds. The edge of the network, which often has modest hardware and memory resources (depending on the network infrastructure provider), can offer vital infrastructure to facilitate DL at the edge. Traditionally to avoid the bottleneck in many mission-critical use cases, edge computing performs tasks such as collection, filtering, and lightweight computation of raw data before transferring data to the cloud [10]. However, with the proliferation of edge servers and progress in DL-based architectures and algorithms, there is a possibility to perform DL model training and deployment efficiently at the network's edge.

The convergence of DL and edge computing has given rise to a new paradigm of intelligence called edge intelligence (EI) [11], [12]. EI aims to facilitate DL deployment closer to the data-generating source. As EI exploits the full potential of resources available at end devices, edge servers, and cloud servers for DL training and inference, based on resource utilization, it is categorized into six levels [13]. These six levels are defined based on where the DL-model training is taking place and where it is getting deployed in the network hierarchy. For simplicity, we assume a network hierarchy formed of cloud servers, edge servers, and end devices. DL training and deployment at cloud servers face significant challenges, including issues such as high latency, data privacy, network congestion, and security threats such as Denial-of-Service attacks [14]. On the other hand, despite being available in huge quantities, end devices suffer from constrained computation power, which is particularly relevant in the context of training and deployment of large DL models. In this regard, edge servers are a viable alternative. Moreover, due to their closer proximity to end devices, edge servers enable reducing network congestion in comparison to the centralized cloud architecture. Furthermore, this proximity minimizes latency, providing for quicker inference when compared with DL models deployed at the cloud server. Even though edge servers have less computing power than the cloud, they do have significantly more computational power than end devices. Thus, edge servers can train and deploy DL models that require larger computing resources than that available at the end devices.

The exclusive use of edge servers for both DL training and deployment is called *all in-edge*. Innovations and research on the emerging area of all in-edge DL processing are in their infancy. Unlike prior surveys [13], [15], [16], [17], [18], [19] summarized in Table 1, to the best of our knowledge, none of the existing surveys presents a detailed view from the all in-edge level perspectives on its enablers, key metrics of performances and challenges when DL is processed at all in-edge level. Specifically, this survey answers the following: leftmargin=0.5cm

1) Which architecture (centralized, decentralized, and distributed) should be used if the configuration of edge servers is known at the all in-edge level?
2) What are the state-of-the-art enabling technologies that facilitate DL training and inference from the all in-edge level?
3) What are the critical performance metrics required in addition to the standard metrics (*e.g.*, accuracy and precision) to evaluate the performance of the DL model's applications at the all in-edge level?

This paper is organized in the following way. It first introduces the computing paradigm and the all in-edge level of EI in Section II. Then, in Section III, it discusses the architecture, enabling technologies for training and inference of DL models at the all in-edge paradigm. Besides, this paper examines the model adaption techniques for effectively deploying DL models at the edge. Next, it reviews the key performance metrics used for evaluating all in-edge DL processing in Section IV. Section V discusses the open challenges and future direction of research for DL at all in-edge. Finally, Section VI presents a summary and identifies the primary conclusions and findings of the paper. Overall, Figure 1 depicts the organization of this paper in the block diagram, and Table 2 provides the list of important acronyms.

II. PRELIMINARY

The centralized nature of the cloud data center has several drawbacks. One of the most considerable disadvantages is the distance between the data centers and end (user) devices, as it requires more wait time to process the data. On the other hand, edge computing offers an indisputable advantage by physically moving storage and processing resources closer to the source of data generation, thereby achieving lower latency. This section presents the distinction between the cloud and edge computing paradigms. Besides, it presents the all in-edge level of the EI paradigm, which comprises only edge servers.

A. INTRODUCTION TO THE CLOUD AND EDGE COMPUTING PARADIGM

The computation of DL can be done by various devices, including cloud servers, edge servers (ESs) and edge

TABLE 1. Summary of related surveys. ✗: Not included; ●: Not considered from all in-edge paradigm; ✔: Included.

Survey Paper	Takeaway	Discussion on computing paradigm	Focused on all in-edge Enablers	Focused on all in-edge Model Adaption	Focused on all in-edge Evaluation Metrics
Edge intelligence: the confluence of edge computing and artificial intelligence [15]	Survey provided insights into edge intelligence. It partitioned edge Intelligence into AI for edge and AI on edge along with research roadmap.	✗	●	●	✗
Convergence of edge computing and deep learning: A comprehensive survey [16]	Survey looked upon EI from machine learning perspective for wireless communication. Also, provides insights into the edge hardware for DL.	●	●	●	✗
Deep Learning with edge computing: A review [17]	Authors of the survey paper looked upon scenarios of EI, techniques to speed up training and inference on ED.	✗	●	●	✗
Wireless network intelligence at the edge [18]	Survey provided insights into theoretical and technical enabler of edge ML for training and inference process.	✗	●	●	✗
Machine Learning at the Network edge: A Survey [19]	Survey looks upon the deployment of ML system at edge of computer along with the tools, frameworks and hardware.	✗	●	●	✗
Edge Intelligence: Paving the Last Mile of Artificial Intelligence With edge Computing [13]	Authors provides six level rating for EI. Survey also provides into the architecture, framework and technologies require for DL deployment over edge.	✔	●	●	●
Enabling All In-Edge Deep Learning: A Literature Review (Ours)	Survey looks upon the key architecture, enabling technologies, model adaption techniques along with performance metric for training and inference from DL for all in-edge level.	✔	✔	✔	✔

TABLE 2. List of important abbreviations.

Abbreviation	Definition
AI	Artificial Intelligence
ANN	Artificial Neural Network
CC	Cloud Computing
CIPAA	Construction Industry Payment and Adjudication Act
DC	Data Center
DL	Deep Learning
DNN	Deep Neural Network
EC	Edge Computing
EDs	Edge Devices
EDGE	Enhanced Data GSM Environment
EI	Edge Intelligence
ES	Edge Server
GDPR	General Data Protection Regulation
IoT	Internet of Things
MEC	Mobile (Multi-Access) Edge Computing
QoS	Quality of Service

devices(EDs). This determines the following computing paradigms.

1) CLOUD COMPUTING

Cloud computing is a paradigm for wide-reaching distributed computing that uses technologies such as grid computing, service orientation, and virtualization. It enables on-demand infrastructure access to a shared pool of configurable computing resources that can be acquired and released with minimum intervention from the server infrastructure provider. Cloud servers have significant storage capacity and computational power to facilitate the overwhelming data coming via the backhaul network from end-user [20], [21]. Thus, cloud servers can satisfy resource requirements for aggregation, pre-processing, and inference for any artificial intelligence-based applications. The cloud servers are inter-connected, providing global coverage with a backhaul network. The cloud computing paradigm involves the end devices that offload data directly to the cloud for further processing. The end devices mentioned here are the originators of the data. In the cloud, data can persist for days, months, and years, meaning long-term temporal data can be collated and processed. For example, cloud data centers facilitate forecasting models based on a large amount of historical time series data [22]. Cloud computing is still the appropriate vehicle for modeling and analytical processing if latency requirements and bandwidth consumption are not an issue, provided measures for preserving privacy and security are in place [23].

2) EDGE COMPUTING

With the surge in the proliferation of IoT devices, traditional centralized cloud computing struggles to provide an acceptable Quality of Service (QoS) level to the end customers [24]. To meet the QoS of IoT applications, there is a need for cloud computing services closer to data sources (*e.g.*, IoT devices, EDs, etc.). As defined by International Electrotechnical Commission (IEC), the extension of computing services from cloud computing to the network edge is called edge computing (EC) [25], [26]. Edge computing helps application developers cater to user-centric services closer to clients. In contrast to cloud computing, latency incurred from edge computing is significantly less, as a majority of data does not have to travel via a backhaul network to the cloud [27]. Less consumption of backhaul networks also means the requirement of bandwidth consumption is considerably less, as shown in Figure 2.

B. ALL IN-EDGE LEVEL OF EDGE INTELLIGENCE

Significant progress has been made in the DL domain in the last decade. Technical advancements in high-performance

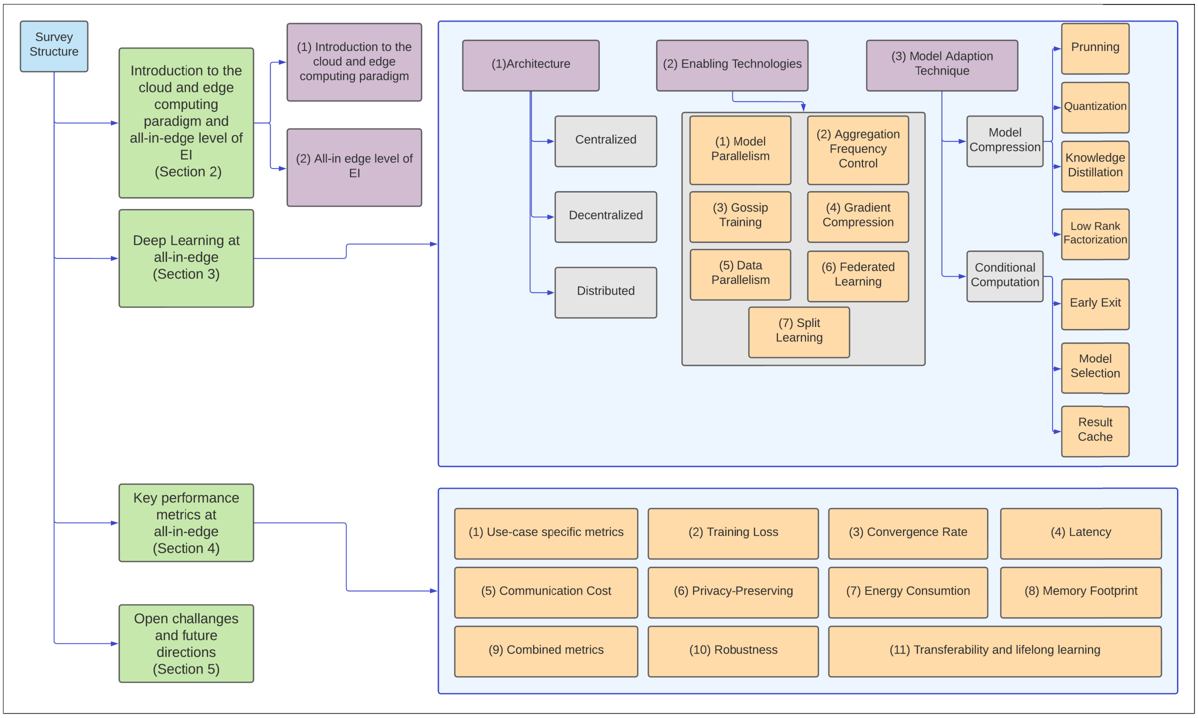


FIGURE 1. Overview of the structure of the survey paper.

processors [28] coupled with improvements in DL algorithms, and the availability and maturity of big data processing [29] have contributed to the increase in DL performance. However, DL processing (training and inference) still occurs mainly in the cloud, as DL models require significant computational resources. As mentioned earlier, this can adversely impact the DL's QoS due to high latency. At the same time, there has been substantial research focused on facilitating DL processing at the edge. While edge computing provides relatively modest computing resources and storage capacity, the training and deployment of DL applications on such devices would greatly help in achieving acceptable QoS for real-time DL applications. For example, real-time applications that would benefit from the merger between edge computing and DL include automated driving [30], and real-time surveillance [31], all of which intrinsically require fast processing and rapid response time [32], [33]. The concept of edge intelligence is a new paradigm that utilizes end devices, edge nodes, and cloud data centers to optimize the processing of DL models (for both training and inference) [13].

As depicted in Figure 2, edge intelligence is divided into six distinct levels based on computational resources offered by the cloud, edge, and end devices for the DL training and inference phase. The fifth level of edge intelligence, depicted in Figure 2, corresponds to all in-edge processing. As defined in [13], all in-edge (fifth level) refers to the edge intelligence paradigm where both training and inference of the Deep Neural Network (DNN) take place in the ES (also known as in-edge manner). This level is critical to satisfying the latency requirements of real-time artificial intelligent applications. In addition, it is helpful in scenarios with intermittent or limited connectivity to the backhaul network [34]. This level helps in reducing the amount of data that needs to be transferred from end devices to the cloud whenever the DL model is being trained. Also, inference provided by the all in-edge level is faster than any other level of EI where inference takes place in the cloud data center [35].

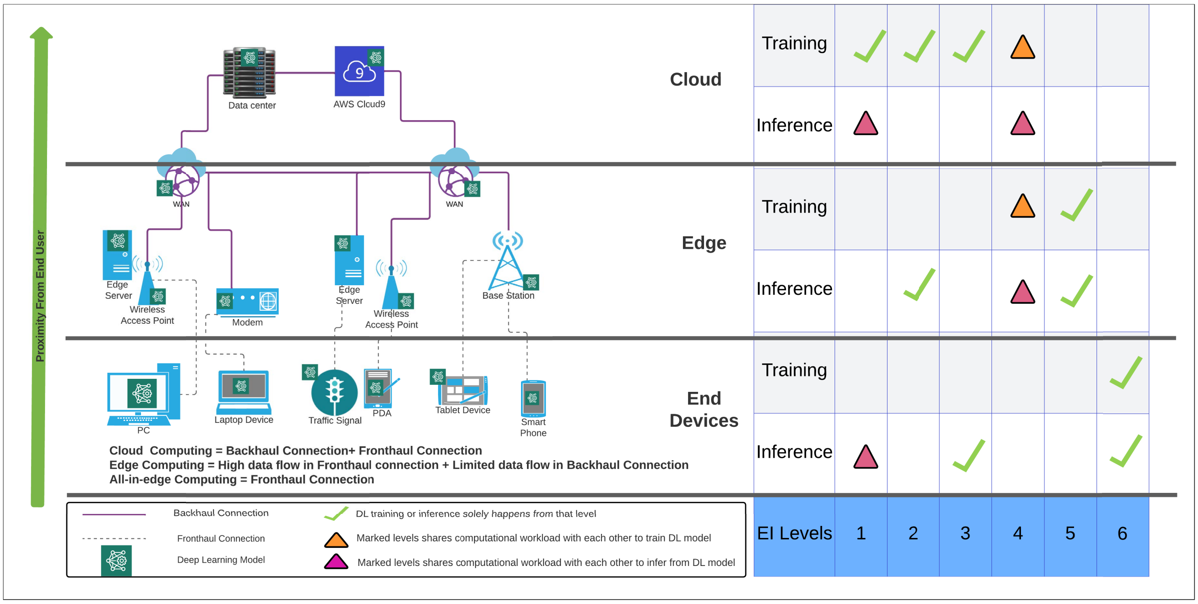
The modest computational resources available with ESs when processing DL models at an all in-edge level facilitate training and inference of relatively large models [36]. Based on the DL model's size, either a single ES can train a DL model or a group of ESs collaborate to train a DL model. Technologies for training DL at level five are described in detail in Section III-B. Similarly, inference from all in-edge can be produced from either a single ES or multiple ESs working collaboratively.

III. DEEP LEARNING AT ALL IN-EDGE

This section reviews the current state of the art for training and adapting DL models from the all in-edge level perspective. Furthermore, the section details the different architectures employed for DL training within the all in-edge level.

A. ARCHITECTURE

The architecture used for DL training at the ES can broadly be divided into three main categories: centralized, distributed, and decentralized, as shown in Figure 3. The architecture


Proximity From End User
Data center
AWS Cloud9
WAN
WAN
Edge Server
Wireless Access Point
Modem
Edge Server
Wireless Access Point
Base Station
PC
Laptop Device
Traffic Signal
PDA
Tablet Device
Smart Phone
Cloud Computing = Backhaul Connection+ Fronthaul Connection
Edge Computing = High data flow in Fronthaul connection + Limited data flow in Backhaul Connection
All-in-edge Computing = Fronthaul Connection
Backhaul Connection
Fronthaul Connection
Deep Learning Model
DL training or inference solely happens from that level
Marked levels shares computational workload with each other to train DL model
Marked levels shares computational workload with each other to infer from DL model

Cloud	Training	✓	✓	✓	△		
	Inference	▲			▲		
Edge	Training				△	✓	
	Inference		✓		▲	✓	
End Devices	Training						✓
	Inference	▲		✓			✓
	EI Levels	1	2	3	4	5	6

FIGURE 2. Layered network architecture with cloud, edge and end devices (the left part of the figure), and the ratings of edge intelligence (EI) into six levels (the right part of the figure).

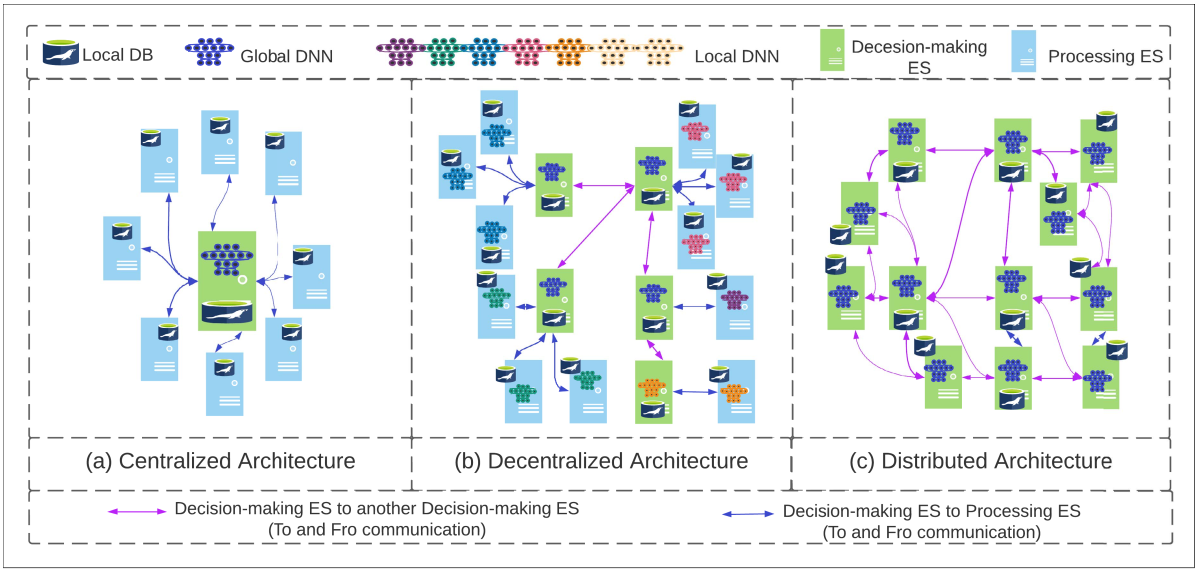

Local DB
Global DNN
Local DNN
Decesion-making ES
Processing ES
(a) Centralized Architecture
(b) Decentralized Architecture
(c) Distributed Architecture
Decision-making ES to another Decision-making ES (To and Fro communication)
Decision-making ES to Processing ES (To and Fro communication)

FIGURE 3. Architecture for training Deep Learning model in-edge: (a) Centralized, (b) Decentralized, and (c) Distributed Architecture.

is defined based on the role of two different types of ES. The first is the processing ES, which is tasked with training the DL model, and the second is the decision-making ES, which coordinates how the model is shared across the network.

1) CENTRALIZED ARCHITECTURE

In a centralized architecture (Figure 3(a)), the processing ES sends the data produced by the end devices (without training local DNN) to the decision-making ES. Decision-making ES then undertakes the DNN training task acting as processing ES [37], [38]. The centralized ES is assumed to have sufficient computing power (and typically, the computing power of the decision-making ES exceeds that of each of the processing ES). In this architecture, the decision-making ES is responsible for acting as both the processing and decision-making ES. Due to decision-making ES acting as processing ES at the same time makes it vulnerable to a single point of failure.

2) DECENTRALIZED ARCHITECTURE

In a decentralized architecture, depicted in Figure 3(b), each processing ES is responsible for training its own local DNN. Once a local model is trained, the ESs send their local DNN model copy to a corresponding decision-making ES. This

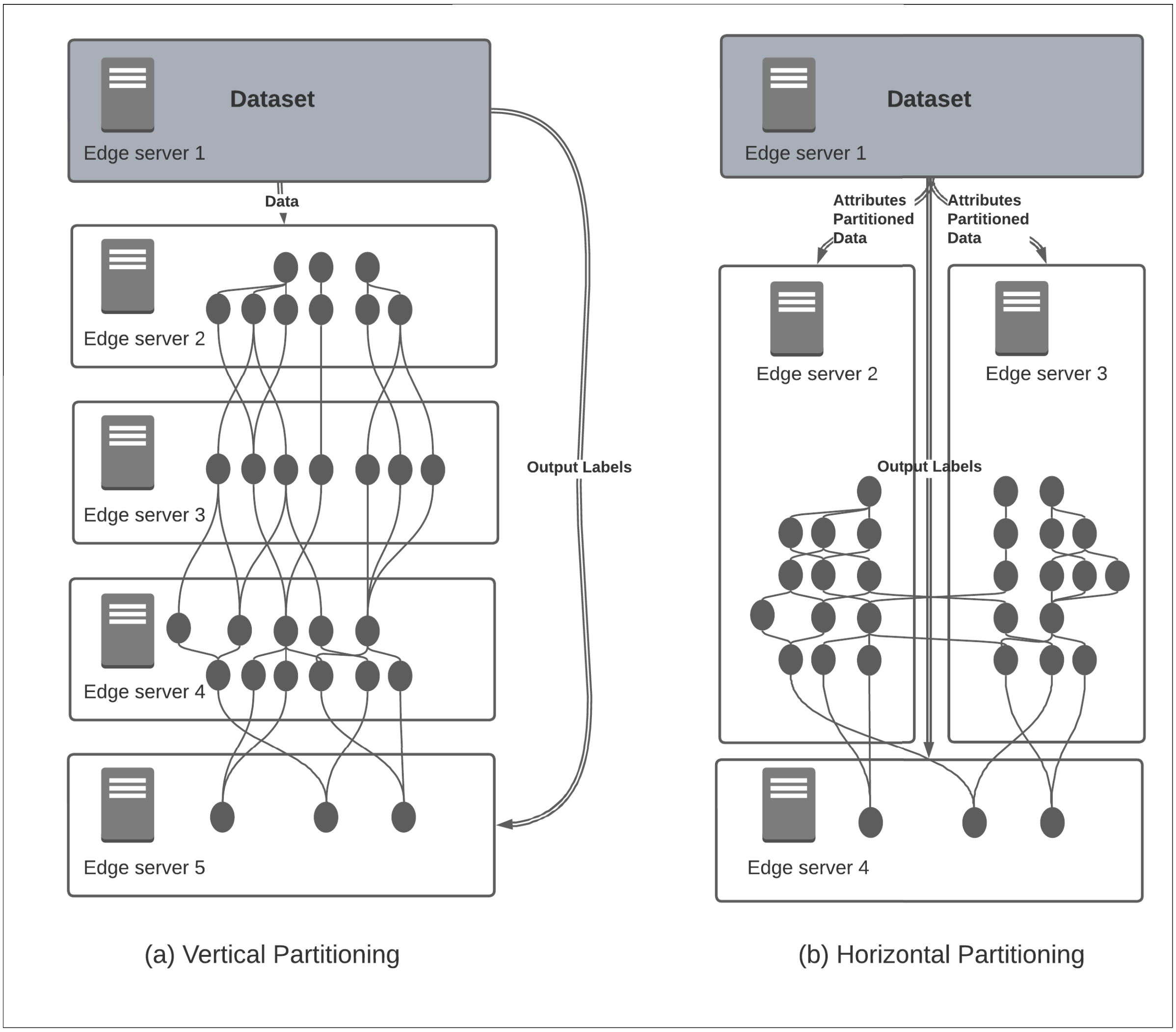


FIGURE 4. Model Parallelism.

decision-making ES aggregates the DNN models and subsequently shares it with other decision-making ES to whom it is connected [40], [41]. Compared to centralized architecture, decentralized architecture addresses the single point of failure by dispersing models amongst multiple decision-making ESs. Thus, even if a single decision-making ES was to go offline, the system could continue operating.

3) DISTRIBUTED ARCHITECTURE

A distributed architecture aims to provide a much more resilient architecture by making each ES (typically decision-making ES) capable of processing (training a local copy of DNN) and making decisions on how to share the data across the networks with other peers. In this architecture, each ES establishes a random peer-to-peer connection with another ES in the network for that specific iteration to share their local models. The receiving ES aggregates received model weights with their local copy of parameters. The training of DNN is stopped once the loss stabilizes in most ESs, and further updating the model parameters does not change the model's estimate for a given classification or regression problem [39].

B. ENABLING TECHNOLOGIES

This section focuses on the technologies that enable the model training process undertaken by the ES. Model parallelism, aggregation frequency control, gossip training, gradient compression, data parallelism, federated learning, and split learning at the ES are emerging technologies, as seen by the substantial amount of research interest and citations, shown in Table 3.

1) MODEL PARALLELISM/DNN SPLITTING

Model Parallelism (also referred to as model splitting or DNN Splitting) is a technique in which the DNN is split across multiple ESs to overcome the constrained computing resources. Model parallelism utilizes a decentralized architecture such that after DNN partitioning, a number of processing ESs train different layers of the DNN model, and a decision-making ES coordinate the training and ensure the correct flow of activations. The model partitioning ensures that the workload assigned to an individual processing ES does not exceed its computational capabilities. Model splitting can be categorized into either horizontally partitioned or vertically partitioned, as shown in Figure 4. In the vertical partitioning approach, one or more layers of the DNN are housed in different servers based on the computational requirement of the layer and the available resources of the processing ES. Whereas in horizontal partitioning, neurons from different layers are placed together based on the computational power of the processing ES. Horizontal partitioning is beneficial when input data is significantly big (number of attributes in a dataset) and single processing ES fails to perform a single-layer operation.

In [40], the authors proposed a framework for scheduled model parallel machine learning called STRADS for vertical partitioned parallel machine learning. The DL application scheduler introduced in the STRADS framework helped control the update of the model parameters based on the model's dependency structure and parameters of the DNN model. The authors also successfully demonstrated 10× faster convergence of the model parallelism-based topic modeling implementation over the model without parallelism. In 2021, research [41] on training the Megatron language model, authors utilized horizontally partitioned model parallelism to train a multi-billion parameter language model. In contrast to the single-GPU-per-model training, the authors in this research implemented model parallelism on the same PyTorch transformer implementations with few modifications. To train such a big system, 512 GPUs were consumed to train the transformer-based model. The same model was then able to achieve the SOTA accuracy on the ReAding Comprehension Dataset From Examinations (RACE [42]) dataset with improved throughput by 10% as compared to existing approaches. Model parallelism provides a way to combine the resources from multiple processing ES to enable all in-edge training of a single DL model.

2) AGGREGATED FREQUENCY CONTROL (AFC)

AFC adopts a decentralized architecture for training DL models, in which a finite number of discrete clusters of ESs are formed, as shown in Figure 5. The task of each of the discrete clusters is to train an identical DNN model. Each cluster has one ES that acts as a decision-making ES. The task of the decision-making ES is to provide all processing ESs in the cluster with an identical copy of the DNN model. Once each processing ES receives its copy, they train that model using their local data and send back the updated DNN model weights to the decision-making ES for aggregation. The decision-making ES aggregates the weights from each of the individual processing ESs in the cluster. Once aggregation is done, the decision-making ES sends back the updated DNN model to all the processing ESs in the cluster. In addition, after each aggregation at the decision-making ES, a "significance function" is computed. This function will determine if the current aggregation has led to a significant improvement. If the improvement is deemed significant, then the current cluster's decision-making ES will inform the decision-making ES of each of the other clusters of the new model weights. Hence, each decision-making ES will have the best available model copy at any given point in time.

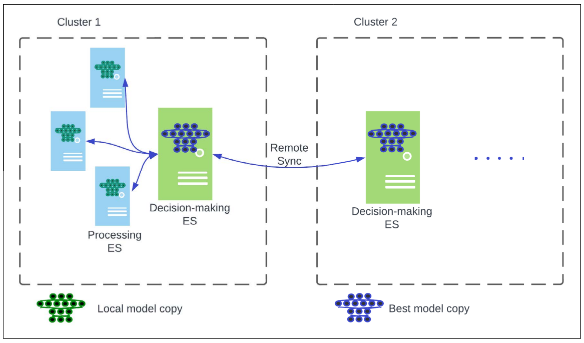


FIGURE 5. Aggregated Frequency Control (AFC).

The significance function in AFC influences the frequency with which updated weights are sent from one decision-making ES to another. This, in turn, can reduce the communication overhead in the network. The Approximate Synchronous Parallel (ASP) model [43] is one such model that targets the problem of geo-distributed DL training. This research successfully employed an intelligent communication system based on the AFC technique achieving minimization in WAN communication by the factor 1.8-53.5× between the two data centers.

3) GOSSIP TRAINING

Gossip Training provides a way to reduce the training time in a distributed architecture. Gossip training is based on the randomized selection of the ES to share the gradient weights for aggregation [44]. Each ES acts as a decision-making ES and processing ES to make the whole training system fault resilient. In this technique, ES will randomly select another node and subsequently send the gradient weight updates to the selected ES. Each ES will then compute the average received weights. Gossip training works in a synchronized and distributed manner. In [45], researchers demonstrated that GoSGD (Gossip Stochastic Gradient Descent) takes 43% less time to converge to the same train loss score when compared to the EASGD (Elastic Averaging SGD [46]) algorithm used in distributed architecture training. In other research, PeerSGD [47] modified the GoSGD algorithm [45] to work in the distributed trustless environment. The algorithm was modified at the stage when the random peer was selected to share the update. The peer who receives the update can decide whether to accept the received weights based on the loss difference (hyper-parameter defined in the research). PeerSGD was evaluated with various clients ranging from 1 to 100. In the experiment, PeerSGD demonstrated 2× faster convergence when tested with 10 clients compared to 100 clients, but it still had comparable accuracy. The limitation of PeerSGD is its inability to achieve convergence in a scenario when data classes are segregated across multiple clients. A modified version of GoSGD was also applied to Wide Area Networks [48], and heterogeneous edge computing platforms [49] and demonstrated results comparable to the original GoSGD algorithm. Gossip training facilitates all in-edge model training without any central authority, making the training process more resilient if any ES is not reachable during training.

4) GRADIENT COMPRESSION

Gradient Compression is another approach to reducing communication while training the DL model, which can be applied to either a distributed or decentralized architecture to facilitate all in-edge training. Gradient compression minimizes the communication overhead incurred by addressing the issue of redundant gradients. Authors in the research [50] found that 99.9% of the gradient exchange in distributed stochastic gradient descent is redundant. They proposed a technique called Deep Gradient Compression, which reduced the communication necessary for training ResNet-50 from 97 MB to 0.35 MB. In gradient compression, two approaches are used in practice: gradient quantization and gradient sparsification.

In gradient quantization [51], gradient weights are degraded from having a higher order of precision values to a lower precision order *i.e.*, representing weights using float 12 rather than float 64. In [52], the author proposed high-dimensional stochastic gradient quantization for reducing the communication in the federated learning setting (federated learning setting is explained in III-B6-6). In the proposed architecture, the authors utilized a uniform quantizer and low-dimensional Grassmannian to decompose the model parameters, followed by compression of the high-dimensional matrix of stochastic gradients into its norm and normalized block gradients. Normalized block gradients are then scaled with a hinge vector to yield the quantized normalized stochastic gradient (QNSD). This QNSD was then transmitted by the processing ES, who trained the model to the decision-making ES, who then aggregates the various gradients and updates a global DL model. Through the framework of hierarchical gradient quantization, authors reduced the communication overhead theoretically and, at the same time, achieved a similar accuracy to the SOTA signSGD scheme [53].

Another approach to gradient compression is gradient sparsification. This technique allows the gradient exchange only if the absolute gradient values are higher than a certain threshold [54]. For example, the threshold in the research ranged from 2 to 15. So if the absolute values of the gradients elements exceed the threshold, they are allowed to be transmitted. The higher the value of the selected threshold, the lower the communication cost (as the threshold limits the transmission of gradient weights). This method reduced the required communication bandwidth by three orders of magnitude for data-parallel distributed SGD training of DNNs. Recent research [55] found that selecting an appropriate threshold is challenging due to the variation in the value of the gradients. This research proposed an alternative approach called the edge Stochastic Gradient Descent (eSGD) method. In eSGD, determining if the gradient update should be sent over the network is based on the loss function. The loss function is used to compute the loss against each coordinate of the gradient at time steps '$t-1$' and 't'. If the loss value at time step 't' is smaller than its value at time step '$t-1$', the current gradient 'gt' will be transmitted to other ESs to build a global model. The standard SGD, when applied to MNIST with 128 batch size and trained for 200000 epochs, will achieve 99.7% accuracy. In contrast, the eSGD method with the same setting attained an accuracy of 95.31% and 91.22% with a drop ratio (% of gradients that will not be communicated by ES) of 25% and 50%, respectively. In [56], the authors aim to identify an optimal trade-off between the communication that takes place within the layers of a DNN (housed in different ESs) and the computations required for the gradient sparsification. The authors developed an optimal merged gradient sparsification algorithm that required 31% less time per iteration over the SOTA sparsified SGD. For the all in-edge paradigm, the size of the message being communicated by the servers utilizes a significant bandwidth. The gradient compression approach helps reduce the size of the message being communicated from one ES to another, thereby freeing up network bandwidth which can then be utilized by other edge applications.

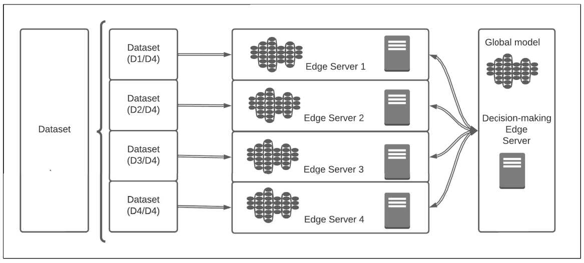


FIGURE 6. Data Parallelism.

5) DATA PARALLELISM

Data parallelism (also referred to as data splitting) is a technique that follows a decentralized architecture at the all in-edge level. A sizeable primary dataset is split in data parallelism to form mutually exclusive smaller datasets. These datasets are then forwarded to the processing ESs. In this architecture (see Figure 6), the decision-making ES initially distributes the uninitialized model copy to each processing ESs. The processing ES starts training after it receives the dataset and the initial model copy. The decision-making ES is responsible for producing the global model by aggregating the local models residing inside the processing ESs. The global model is next sent back to the processing ESs so that it can continue to update its local model [57], [58], [59].

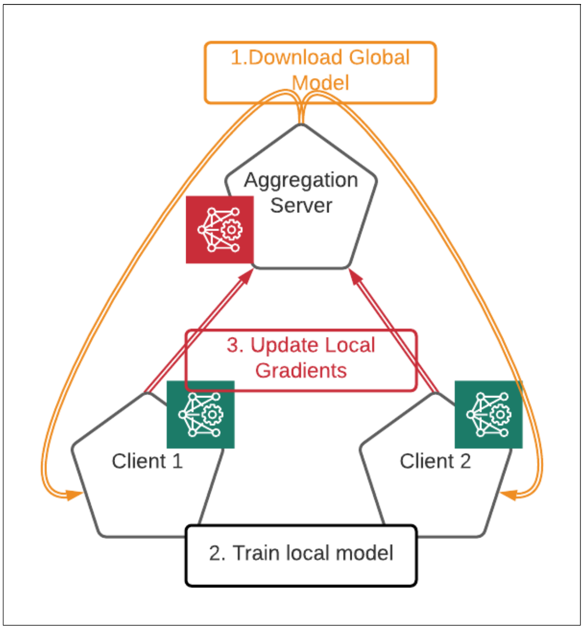


FIGURE 7. Vanilla federated learning.

6) FEDERATED LEARNING

Federated learning (FL) is a popular framework for training DL models using a decentralized and distributed architecture [60]. Although the native framework treats mobile devices as clients responsible for training the DL model, recent research shows clients can be extended to the ES [61], [62], which makes this technology applicable for all in-edge. In this section, the 'client' refers to processing ES with low computing resources, and the 'aggregation ES' refers to decision-making ES with modestly higher computing capacity than the client.

Federated learning enables ES to collaboratively learn a shared DL model while keeping all the training data on the client. As shown in Figure 7, during the first stage, all the clients download the global DL model from the aggregation ES, which is responsible for maintaining the global DL model. Once the global DL model is received, the client trains it using its own private data, making it a local DL model. Once training is completed on the client, the local model weights are sent to the aggregation ES. Once the aggregation ES receives all the weights from the participant client, it is then aggregated to formulate the new global DL model [63], [64], [65]. After aggregation, the global DL model is again circulated to the client for further training, making the whole approach cyclic. This framework ensures that the performance of the aggregated global model should be better than any of the individual client-side models [66] before being disseminated.

Federated Learning Systems (FLS) can be further categorized based on their data partitioning strategy, privacy mechanism, and communication architecture [66], [67], [68], [69]. The data partitioning strategy dictates how the data is partitioned across the clients. There are three broad categories of data partitioning (i) horizontal data partitioned FLS, (ii) vertical data partitioned FLS, and (iii) hybrid data partitioned FLS. In horizontal data partitioning, all the clients have the same attributes/features in their respective datasets needed to train the private DL model. Whereas in vertical data partitioned, all the clients have different attributes/features in the dataset. By utilizing entity alignment techniques (which helps find the overlap in other datasets where some of the features are common) [70], [71], overlapped samples are collected for training machine learning models. Hybrid data partitioning utilizes the best of both worlds. The entire dataset is divided into horizontal and vertical subsets in this category. So each subset can be seen as an independent dataset with fewer non-overlapping attributes and data points compared to the entire dataset [68].

FLS provides privacy to a certain degree by default by allowing raw data to stay only with the client ES. However, while exchanging the model parameters, there is the possibility that exchanged model parameters could still leak some sensitive information about private data [72]. Therefore, privacy mechanisms have been employed for FLS. These mechanisms can be subdivided into either cryptographic techniques or differential privacy techniques. Cryptographic techniques require that both the client and aggregation ES operate on encrypted messages. Two of the most widely used privacy-preserving algorithms are homomorphic encryption [73], [74], [75], [76] and multi-party computation [77], [78], [79]. On the other hand, differential privacy introduces random noise to either the data or the model parameters [80], [81], [82], [83]. Although random noise is added to the data or model parameters, the algorithm provides statistical privacy guarantees while ensuring that the data or model parameters can still be used to facilitate effective global model development.

The communication architecture of an FLS can be broadly subdivided into two subcategories: distributed and decentralized architectures. In a decentralized architecture, the aggregation server is responsible for collecting and aggregating the local models from each client. It then sends the updated global model for retraining to each client. In this architecture, communication between the processing ES and decision-making ES can happen in synchronous [68], [84] as well as in asynchronous [84], [85], [86], [87] manner. One of the significant risks in a decentralized architecture setting is that the decision-making ES may not treat each processing ES equally. That is, the decision-making ES may have a bias toward specific processing ES due to their higher participation during a training phase. A distributed architecture can mitigate the potential issues of bias. A distributed architecture in federated learning can be based on a P2P scheme (ex., gossiping scheme as described in Section III-B3), a blockchain-based system, or a graph-based system. In a distributed architecture, all the participating ESs are responsible for acting as processing and decision-making ES. Therefore, if a gossip scheme is implemented to achieve the decentralized FLS, all the models will randomly share the updates with

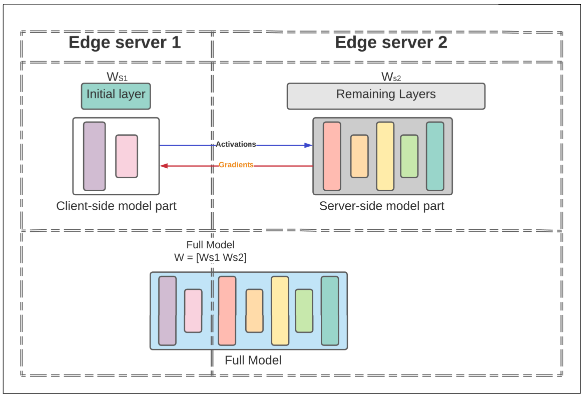


FIGURE 8. Split Learning.

their neighbors [88], [89]. In contrast, if a blockchain system is implemented, it leverages smart contracts (SC) to coordinate the DL training, model aggregation, and update tasks in FLS [90], [91], [92], [93], [94]. Lastly, if graph-based FLS is implemented, each client will utilize the graph neural network model with its neighbors to formulate the global models [95], [96], [97], [98].

FLS provides a much-needed way of enabling the DL model training and inference at the all in-edge paradigm. With an FLS, one can easily integrate multiple low-resource ESs to help train the DL model at the edge. Also, based on the resources available at the edge and the communication overhead of FLS, one gets the freedom to select either a distributed or decentralized architecture.

7) SPLIT LEARNING

In federated learning, each processing ES is responsible for locally training the whole neural network. In contrast, split learning provides a way to offload some of this computation between processing and decision-making ESs. More differences between federated learning and split learning are summarized in Table 3. As we advance in this section, the 'client' refers to processing ES with low computing resources, and the 'server' refers to the decision-making ES with a modestly higher computing capacity than the client. Split learning divides a neural network into two or more sub-networks. Figure 8 illustrates the case where we split a seven-layer neural network into two sub-networks using layer 2 as the "cut layer". After the split, the two sub-networks are shared between the client, who trains the initial two layers of the network, and the server, who trains the last five layers of the network. At the training time, the client initiates the forward propagation of its confidential data and sends the activation from the cut layer to the server-side sub-network. The server then continues the forward propagation and calculates the loss. During backpropagation, gradients are computed and propagated initially in the server sub-network and then relayed back to the client side sub-network. In Split learning, during the training and testing, the server never gets access to the parameters of the client-side network or the client's data.

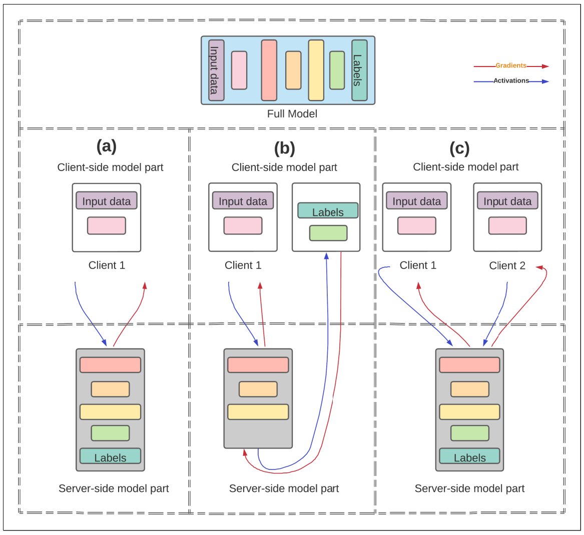


FIGURE 9. Different configurations of Split Learning- (a) simple vanilla split learning, (b) split learning without label sharing and (c) split learning for vertically partitioned data.

Split learning can be broadly categorized into three configurations based on how the input data and labels are shared across the clients and servers. Figure 9 shows three configurations- simple vanilla split learning, split learning without label sharing, and split learning for vertically partitioned data. A main neural network is partitioned into two sub-networks in simple vanilla split learning. The initial sub-network, along with the input data for the neural network, remains with the client, whereas the remaining sub-network, along with the labels, resides with the server [99]. Split learning without label sharing is identical to vanilla split learning, except that the labels reside with the client instead of the server. To compute the loss, the activations outputted from the server-side network are sent back to the client, who holds the last layer of neural network [100]. The loss is calculated, and gradients are computed from the last layer held by the client and then sent back to the server, and backpropagation takes place in the usual way. The final configuration of split learning is where the clients train their partial sub-network for vertically partitioned data and then propagate the activations to the server-side sub-network. The server-side sub-network then concatenates the activations and feeds them to the remaining sub-network. In this configuration, labels are also shared with the server [101].

In a federated learning system, clients can interact with the server in parallel, which helps achieve faster training compared to a split learning approach. In split learning, the server must wait for all clients to send their activations before propagating the activation through the server-side network. Also, in contrast to federated learning, split learning reduces the computational requirements on the client-side (as only a partial amount of the network resides with the client). Recently, to leverage the advantages of both split learning and federated learning, a hybrid technique called splitfed learning was proposed [102].

TABLE 3. Comparison of enabling techniques for all in-edge.

Category	Model Parallelism/ DNN Splitting	Aggregation Frequency Control	Gossip Training	Gradient Compression	Data Parallelism	Federated Learning	Split Learning
Partial model training	Yes	No	No	No	No	No	Yes
Parallel model training	No	Yes	Yes	Applicable for parallel/ non-parallel model training.	Yes	Yes	No
Model	Single DNN model is partitioned across multiple ESs.	Multiple clusters of ESs collaboratively train the model.	Multiple ESs train model collaboratively.	Helps to reduce the size of gradients passed across ESs during collaborative training.	Large dataset is split and passed to multiple ESs to train the model collaboratively.	Clients server collaboratively trains the model.	Client server shared model architecture and collaborative training.
System Architecture	Decentralized	Decentralized	Distributed	Decentralized/ Distributed	Decentralized	Decentralized/ Distributed	Decentralized/ Distributed
Inter-communication during model training	Only activation vectors from the last layers are shared.	Full model parameters exchanged.	Full model parameters exchanged.	Compressed model parameters exchanged.	Full model parameters exchanged.	Full model parameters exchanged.	No model parameters are exchanged (only activation vectors from the last layers are shared)
Computational resource requirements for large DNN	Equal computation resources are required at each Ess.	High at client and server end.	Equal computation resources are required at each ESs.	Equal computation resources are required at each ESs.	Equal computation resources are required at each ESs.	High at client and server end.	Low at the client end and comparatively high at the server end.
Data privacy by default	No	Yes	Yes	No	No	Yes	Yes
Communication overhead between server and clients	Depends on a number of partitions.	Depends on model size.	Depends on model size.	Reduced	Depends on the number of partitions.	Depends on model size.	Depends on sample size and the number of nodes in the cut-layer.
Related Research	[40, 41, 104, 105, 106, 107]	[43]	[39, 108, 109, 110]	[51, 54, 55, 111, 112]	[59, 113, 114, 115, 116, 117]	[63, 64, 65, 66, 67, 68, 69, 71, 118, 119]	[99, 100, 102, 103, 120]

In splitfed learning, a DL model is broken down into the sub-networks shared amongst the clients and servers. In addition, there is a separate federated aggregation server for the client and the servers. All the clients perform the forward pass in parallel and independently of each other (as not seen in split learning). The resulting activations are sent to the server-side sub-network, which performs a forward pass for the remaining sub-network portion. The server then calculates the loss and back propagates the gradients back to the very first layer on the client-side, as described earlier with split learning. Once this process finishes, the servers send their model weights to a federated aggregation server, which aggregates the independent server-side sub-network to form a global server-side model. Similarly, the clients send their sub-network weights to another aggregation server. At the end of aggregation, a global model can be developed by combining the aggregated client-side weights with the aggregated server-side weights as shown in Figure 10 (a) [102], [103].

Splitfed learning can have several variants. For example, the first one is where each client has its own corresponding server-side network in the main server, *i.e.*, the number of client-side models is equal to the number of server-side models as explained in the earlier paragraph. In the second variant, there are multiple clients but only a single server. Therefore, each client-side model sends its activations to a single common server-side sub-network, thereby reducing the required aggregation step and the need to keep multiple copies of the server-side networks as compared to the first variant as shown in Figure 10 (b). Moreover, as the server keeps only one copy of the server-side sub-network, it makes the server-side do forward and backward pass sequentially with each of the client's data (activations of the cut layer) [103], [121].

Key takeaways: The above-mentioned enabling technologies at the confluence of DL and all in-edge contribute to our understanding of training DL models using only ESs. The enabling technologies help address issues such as limited computational resources, communication overhead and latency between ESs, data privacy, and model robustness. Model parallelism and split learning provide a means of decreasing the computational resource required by individual ES. By splitting the DL model, multiple resource-constrained ESs can train a few layers of the network (rather than the entire model). Aggregated frequency control and federated learning enable parallel model training, facilitating faster model convergence. Gossip training, federated learning, and aggregated frequency control adopt a distributed architecture, thereby robustly training a DL model in situations where the reliability of an ES is not predictable. Also, we have discussed splitfed, in which federated learning is combined with split learning. Splitfed overcomes the drawback of federated learning of training a large ML model in resource-constrained ESs [122]. At the same time, it eliminates the weakness of split learning to deal with one client at a time while training [121].

C. ALL IN-EDGE MODEL ADAPTION

Model Adaption techniques provide a means by which DL model deployment at the ES can be achieved despite the lack of computing resources, storage, and bandwidth. Model adaption techniques can be broadly categorized into model compression and conditional computation techniques, as summarized in Table 4.

1) MODEL COMPRESSION

Model compression techniques facilitate the deployment of resource-hungry DL models into resource-constrained ES by reducing the number of parameters or training DL models that have been reduced in size from the original model. Model compression exploits the sparse nature of DL models by compressing the model parameters. Model compression reduces the computing, storage, memory, and energy requirements needed for all in-edge deployment of DL models. This section reviews pruning, quantization, knowledge distillation, and low-rank factorization.

a: PRUNING

Pruning of parameters is the most widely adopted approach to model compression. This approach evaluates DL model parameters against their contribution to predicting the label. Those neurons that make a low contribution in inference are pruned from the trained model. Parameter pruning can significantly reduce the size of a DL model, but it also has the potential to impact the model's performance adversely. In [12], the authors were able to reduce the size of the AlexNet and VGG-16 by a factor of 9× and 13× respectively, without incurring any loss in the accuracy over the ImageNet dataset. In another work [123], the authors utilized pruning to create a compressed speech recognition model on field-programmable-gate-array (FPGA). This technique compressed the LSTM model by 10× with negligible loss in accuracy. SS-Auto [124] is a single-shot structured pruning framework. In contrast to earlier versions of pruning where the entire DL model's parameters were selected for pruning, in structured pruning, independent pruning on columns and rows of filters and channels matrix (for CNN-based DL models) is performed. The compressed DL models produced by the SS-Auto framework did not suffer any degradation in performance, achieving the original performance levels when tested on CIFAR-10 and CIFAR-100 datasets. However, the compressed VGG-16 model reduced the number of convolutional layers parameters by a factor of 41.4% for CIFAR-10 and 17.5% for the CIFAR-100 dataset. In [125], the authors proposed a new framework based on weight pruning and compiler optimization for faster inference while preserving the privacy of the training dataset. This approach initially trains the DL models as usual on the user's data. The model then undergoes privacy-preserving-oriented DNN pruning. Finally, synthetically generated data (with no relevance to the training data) is passed through a layer of the user-trained model. The decision to prune a parameter

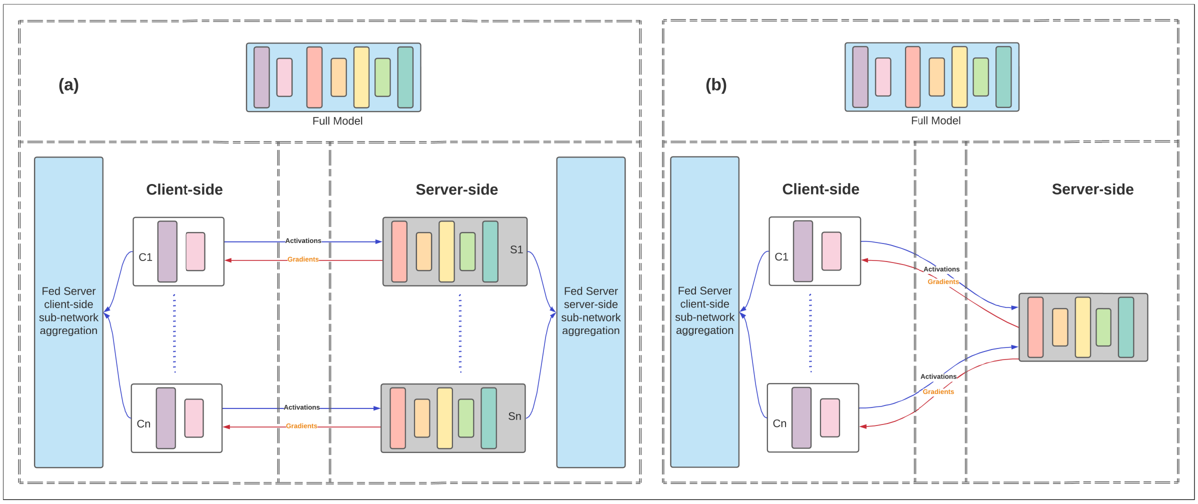

(a)
Full Model
Client-side
Server-side
Fed Server client-side sub-network aggregation
C1
Cn
Activations
Gradients
S1
Sn
Fed Server server-side sub-network aggregation
(b)
Full Model
Client-side
Server-side
Fed Server client-side sub-network aggregation
C1
Cn
Activations
Gradients

FIGURE 10. Variants of splitfed learning (a) Splitfed learning with the same number of client and server-side sub-networks and (b) Splitfed learning with only one copy of server-side sub-network.

or not from the current layer is based on how similar (by computing the Frobenius norm) the original output of the layer (without pruning) is when compared with the output of the layer after the parameter has been pruned. If the outputs are close enough, then that parameter is pruned. This pruning technique is named the alternating direction method of multipliers (ADMM). Experimental results of the framework outperformed the state-of-the-art end-to-end frameworks, i.e., TensorFlow-Lite, TVM, and MNN, with speedup in inference up to 4.2×, 2.5×, and 2.0×, respectively.

b: QUANTIZATION

Data quantization degrades the precision of the parameters and gradients of the DL model. More specifically, in quantization, data is represented in a more compact format (lower precision form). For example, instead of adopting a 32-bit floating-point format, a quantization approach might utilize a more compact format such as 16-bit to represent layer inputs, weights, or both [13]. Quantization reduces the memory footprint of a DL model and its energy requirements. In contrast, pruning the neurons in a DL model will reduce the network's memory footprint but does not necessarily reduce energy requirements. For example, if later-stage neurons are pruned in a convolutional network, this will not have a high impact on energy because the initial convolutional layer dominates energy requirement [13]. In [126], the authors utilized a dynamic programming-based algorithm in collaboration with parameter quantization. With the proposed dynamic programming-assisted quantization approach, the authors demonstrated a 16× compression in a ResNet-18 model with less than a 3% accuracy drop. The authors in [127] proposed a quantization scheme for the inference phase of the DL model that targets weights along with the inputs to the model and the partial sums occurring inside the hardware accelerator. Experiments showed that the proposed schema reduced the inference latency and energy consumption by up to 3.89× and 4.84×, respectively, while experiencing a 1.18% loss in the DL models inference accuracy.

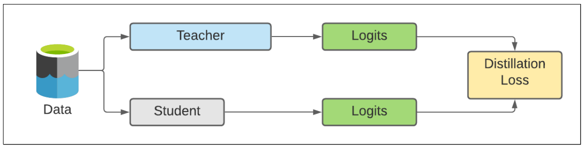


FIGURE 11. Teacher-student architecture for Knowledge Distillation.

c: KNOWLEDGE DISTILLATION

Knowledge distillation is a model compression technique that helps train a smaller DL model from a significantly larger trained DL model. The knowledge distillation comprises three key components: (i) The original knowledge, (ii) the distillation algorithm, and (iii) the teacher-student architecture [128]. The original knowledge is the original large DL model, which is referred to as the teacher model. The knowledge distillation algorithm is used to transfer knowledge from the teacher model to the smaller student model using techniques such as Adversarial KD [129], [130], Multi-Teacher KD [131], [132], [133], Cross-modal KD [134], [135], Attention-based KD [136], [137], [138], [139], Lifelong KD [140], [141] and Quantized KD [142], [143]. Finally, the teacher-student architecture is used to train the student model. A general teacher-student framework for Knowledge distillation is shown in Figure 11. In this architecture, the teacher DL model is trained on the given dataset in the initial phase. Once the teacher DL model is trained, it assists the shallower student DL model. The student DL model also uses the same dataset used to train the teacher DL model, but labels for the data points are generated by the teacher DL model [144]. The knowledge distillation technique helps a smaller DL model imitate the larger DL model's behavior.

KD provides a viable mechanism of model compression [128]. This technique helps reduce the number of ESs required to deploy the larger DL model at the all in-edge level. Reduction in the number of ES also helps achieve faster inference time from ESs (as less communication needs to be done within ESs).

d: LOW-RANK FACTORIZATION

Low-rank factorization is a technique that helps in condensing the dense parameter weights of a DL model [145], [146], limiting the number of computations done in convolutional layers [147], [148], [149] or both [150], [151]. This technique is based on the concept of creating another low-rank matrix that can approximate the dense metrics of the parameter of a DL model, convolutional kernels, or both. Low-rank factorization can save memory on an ES while decreasing computational latency because of the resulting compact size of the DL model. In [152], the authors used the low-rank factorization by applying a singular value decomposition (SVD) method. They demonstrated a substantive reduction in the number of parameters in convolutional kernels, which helped reduce floating-point operations(FLOPs) by 65.62% in VGG-16 while also increasing accuracy by 0.25% when applied to the CIFAR-10 dataset. Unlike pruning, which necessitates retraining the DL model, after applying low-rank factorization, there is no need to retrain the DL model. Further research [153] proposed a sparse low-rank approach to obtain the low-rank approximation. The sparse low-rank approach is based on the idea that all the neurons in a layer have different contributions to the performance of the DL model. So based on the neuron ranking (based on the contribution made for inference), entries in the decomposition matrix were made. This approach, when applied over the CIFAR-10 dataset with VGG-16 architecture, achieved 3.6× times smaller compression ratio to the SVD. Other commonly used methods for low-rank factorization are tucker decomposition (TD) [154], [155], [156] and canonical polyadic decomposition (CPD) [157], [158].

2) CONDITIONAL COMPUTATION

Conditional computational approaches alleviate the tension between the resource-hungry DL model and the resource-constrained ES. In conditional computation, the computational load of the DL model deployed over a single ES is distributed with other ES in the network. The selection of an appropriate conditional computation technique is based on the DL model's latency, memory, and energy requirements. Therefore, depending upon the configuration of the ES and DL model's computation requirements, DL model deployment can utilize one or any combination of the techniques (Early Exit, Model Selection, and Result Cache) defined in this section.

a: EARLY EXIT

The main idea behind the early exit approach is to find the best tradeoff between the deep DNN structure of a DL model and the latency requirements for inference. In this approach, a deep neural network trained on a specific task is partitioned across multiple ESs. The partitioning of the DL model is based on a layer-wise split, such that a single or multiple layers can reside across multiple ESs based on the computation power provided by each ES. Each ES that hosts one or more layers of the DL model also attaches a shallower model (or side branch classifier) to the output of the final layer on the current ES. The model is then trained as shown in Figure 12. The purpose of the side branch classifier is to provide an early prediction or early exit. During inference, the data is propagated through the network (and each ES host). Each host will calculate both the output of the hosted layers and the output of the local early exit network. If the output of the early exit layer exceeds a defined confidence threshold, then the propagation stops (this is the early exit), and the 'early' result is returned. If the prediction from the early exit network is less than the confidence threshold, the output of the larger DL model's layers is then propagated to the next ES in the chain, which holds the next layer of the larger DL model and another early exit network. The process of propagating the layer's output to the subsequent layer is carried out until one ES inferences the class with a higher confidence score. This process can provide '$n-1'$ exit points for a DL model with 'n' neural network layers; thus, if layer 1 of the larger DNN along with the side branch can infer the class with the required confidence that output will be given as a response to the end user eliminating any further propagation of activation values along the ES.

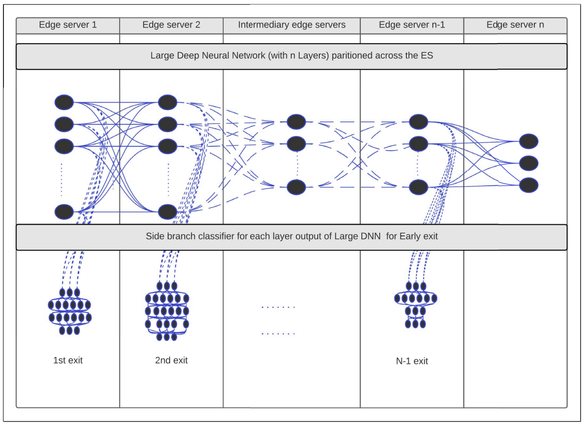

Edge server 1
Edge server 2
Intermediary edge servers
Edge server n-1
Edge server n
Large Deep Neural Network (with n Layers) paritioned across the ES
Side branch classifier for each layer output of Large DNN for Early exit
1st exit
2nd exit
N-1 exit

FIGURE 12. Early exit adaption of Deep Neural Network.

Researchers in [159] provided the programming framework 'Branchynet', which helps incorporate the early exit approach into a standard DL model. The framework modifies the proposed DL model by adding exit branches at certain layers. With the multiple early exit points, it can also be considered as an enabler for localized inference using DL models with less number of layers. For the AlexNet DL model, 'Branchynet' framework was able to reduce the inference time by a factor of 2× and 6× on CPU and GPU, respectively. In [160], the authors proposed DeepQTMT to lower the encoding time spent on video compression. In the DeepQTMT, the authors utilized a multi-stage early exit mechanism to reduce the high encoding time. Experimental results showed the encoding time was reduced by a factor ranging from 44.65% to 66.88% with a negligible adverse impact in bit-rate within the range of 1.32% to 3.18%. Therefore, while the early exit strategy can decrease latency and facilitate a faster response time, it does have the drawback of increasing the memory footprint of the DL model, thus utilizing more storage at each ES.

b: MODEL SELECTION

The model selection approach selects a specific DL model for inference from a set of available DL models based on the latency, precision, and energy requirements of the end user [16]. In a model selection strategy, multiple DL models with varying DL model structures are trained. The different trained models each have a specific inference latency, energy requirements, and accuracy. Once trained, each of the models is deployed to various servers. The model selection approach will then select the DL model based on the end user requirements [13].

The model selection approach is similar to the early exit approach, with only one difference. In model selection, independent DL models are trained; in contrast, in the early exit, only one DL model is trained over which multiple exit points are created. Authors in [161] proposed a new concept of BL-DL (big/little DL) based on the model selection approach. The authors proposed the score margin function, which helps in deciding whether or not the inference made by a small DL model is valid. The score function is computed by subtracting the highest probability from the second-highest probability of a class from the last classifier layer of a DL model. Thus, a score function can be seen ranging from 0 to 1. The higher the value of the score function, the higher the estimation that inference is accurate. The lower the value of the score function, the lower the estimation of inference being accurate. If the score function estimation is low, then a larger DL model is invoked to make the inference on the same input data. The same research showed a 94.1% reduction in the energy consumption on the MNIST dataset, with accuracy dropping by 0.12%. Recently in [162], an adaptive model selection technique has been used to optimize the DL model's inference. The proposed framework builds a standard DL model, which learns to predict the best DL model to use for inference based on the input feature data. To facilitate the training of the selection model (which is the standard KNN model in this scenario), different pre-trained models like Inception [163], ResNet [164], MobileNet [165] were evaluated on the same image dataset. For each image, the DL model that achieved the highest accuracy is set as the output. The training data for the KNN model comprises the features extracted from the image as input and the optimal DL model as output. Once the model selector (the KNN) is trained, it is then used to determine the DL model, giving the best accuracy on the selected image. In the end, the selected DL model makes an inference on the image as shown in Figure 13.

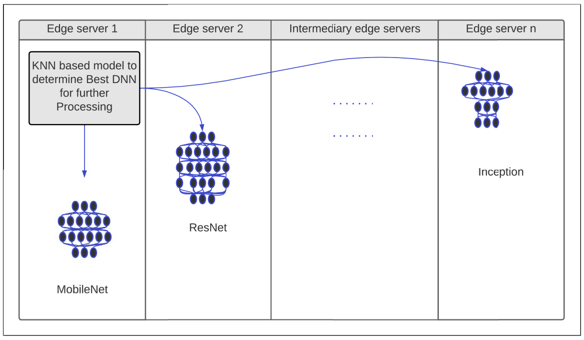

Edge server 1
Edge server 2
Intermediary edge servers
Edge server n
KNN based model to determine Best DNN for further Processing
MobileNet
ResNet
....
....
Inception

FIGURE 13. Model Selection of Deep Neural Network.

Experimental results validated the reduction in the inference time by a factor of 1.8× for the classification task and 1.34× time reduction in a machine translation task. While model selection facilitates a decrease in inference time, it does incur an increased memory footprint across the ESs due to the number of pre-trained DL models.

c: RESULT CACHE

Result cache techniques help in decreasing the time required to obtain the prediction from the ES. In this approach, frequent input queries (such as frames in the case of video classification or images in the case of image classification) and the associated predictions made by the DL model are saved in an archive on the ES. So, before any query is inferred from the DL model, intermittent lookup happens. In intermittent lookup, if a query is similar to a saved query, the result is inferred from the archive (cache). Otherwise, the query goes to the DL model for inference. This technique becomes more powerful in environments where the queries can be expected to exhibit similarity. In [166], the authors proposed a cache-based system that leveraged the ES for image classification. When evaluated on image classification applications, the approach yielded up to 3× speedup on inference for image recognition tasks without any drop in the model's performance (accuracy). Another system for video analysis utilized the cached convolution outputs of the CNN layers to reduce the computation for making an inference [167]. The idea is again based on the similarity of consecutive frames in videos. Initially, in this approach, activations from each layer of DL for a query frame are saved in the cache. For the next subsequent frame (query), the query is pushed through the first layer, and the resulting activations are compared with the previous activation values of the same layer saved in the cache. Only those activations that differ significantly from the cached version are calculated and propagated further through the network. If the activation is deemed similar, they are carried over with their cache results to the next layer. In the experiment, the authors showed a significant speedup of 3× to 4× compared to the vanilla CNN model with no change in accuracy. In other research [168], the authors proposed a framework similar to result caching. In this research, queries were initially passed through the DL model, and activations of each layer were cached (archived) in the ES along with the prediction from the DL model. During the inference, after

TABLE 4. Comparison of model adaption techniques for all in-edge.

Category	Pruning	Quantization	Knowledge Distillation	Low rank factorization	Early exit	Model selection	Result Cache
Model adaption category	Model compression	Model compression	Model compression	Model compression	Conditional computation	Conditional computation	Conditional computation
Number of DL models involved	Single	Single	Single	Single	Multiple	Multiple	Single/ Multiple
DL computation	Reduced	Reduced	Reduced	Reduced	It can be reduced or increased based on exit criteria.	Depends on the DL model selected for making inferences.	No impact
DL model size	Reduced	Reduced	Reduced	Reduced	Increased	Depends on DL model selection technique.	No impact
Accuracy drop due to model adaption technique	Yes	Yes	Yes	No	No	No	No
Memory footprint	Reduced	Reduced	Reduced	Reduced	Increased	Increased	Increased
Inference time	Reduced	Reduced	Reduced	Reduced	It can be reduced or increased based on exit criteria.	Depends on the DL model selected for making inferences.	Reduced
Related Research	[12, 123, 124, 125, 169, 170, 171, 172, 173]	[13, 126, 127, 170, 171, 174, 175]	[129, 130, 131, 132, 133, 135, 136, 137, 139, 140, 141, 176, 177, 178, 179, 180]	[145, 146, 147, 148, 149, 150, 151]	[16, 159, 160, 181, 182, 183, 184, 185]	[13, 16, 161, 162]	[166, 167, 168, 186, 187, 188, 189, 190, 191, 192]

passing the image through the layers of DL, activations are checked with the saved activations of a specific layer. If the activations of a particular layer for the current query match with the activations in the cache, further propagation of the activations is stopped, and the cached result is returned as the prediction. This research was applied to a VGG-16 architecture using CIFAR and yielded a 1.96× latency gain using a CPU and a 1.54× increase when using a GPU with no loss in accuracy. Result caching provides a significant boost in the scenario where the query (frames processing for the boundary identification) for inference does not change significantly. While result caching improves the overall latency of the neural network, it also incurs a larger memory footprint.

Key takeaways: This section described model adaption techniques, which facilitate the efficient deployment of large DL models at the all in-edge level, divided into segments- model compression and conditional computation, summarized in Table 4.

Model compression techniques such as pruning, quantization, knowledge distillation, and low-rank factorization provide practical ways of reducing the size and memory footprint of the DL model. Reduction in model size due to model compression techniques also decreases the amount of computation needed for making an inference. However, there lies a need for DL model retraining while adopting pruning and knowledge distillation, whereas no retraining is required for quantization and low-rank factorization. Also, a drop in accuracy is observed amongst all model compression techniques leaving low-rank factorization.

Conditional computation techniques such as early exit, model selection, and result caching provide practical ways to utilize the computational resources of the available ESs to provide faster inference. In contrast to a reduced memory footprint observed while using the model compression technique, the memory footprint increases while adopting conditional computation. Also, no significant drop in accuracy is observed while utilizing the conditional computation techniques.

IV. KEY PERFORMANCE METRICS OF ALL IN-EDGE

The application of DL at the edge has gathered significant momentum over the last few years. Typically research evaluates the performance of a limited number of DL models often adopting a different set of standard performance metrics (such as top-k accuracy [193] and mean average precision [194]). Unfortunately, these standard metrics fail to provide insights into the runtime performance of DL model inference at ESs. Relevant performance metrics for DL services include but are not limited to latency, use-case-specific metrics, training loss, communication cost, privacy-preserving metrics, energy consumption, memory footprint, combined metrics, robustness, transferability, and lifelong learning. Table 5, summarizes the metrics/ description against KPI utilized at all in-edge.

This section will discuss the different metrics that should be evaluated when developing all in-edge based DL models.

A. USE-CASE SPECIFIC METRICS

Use-case specific metrics are used to determine the quality of the trained DL model and are dependent on the problem statement. For example, if the use-case is a classification problem, then accuracy, F1-score, roc_auc, etc. can be evaluated [195], [196], [197]. Accuracy and F1-score are the most common metrics used to determine the quality of classification problems. In the classification problem, the DL model is trained to correctly predict the class of interest, i.e., true positive (TP) and the class of dis-interest, i.e., true negative (TN). Equation 1 represents the mathematical formulation of Accuracy, where FP is a false positive (classes that are wrongly classified as positive), and FN is a false negative (classes that are wrongly classified as negative).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{1}$$

Equation 2 denotes the calculation of F1-score, commonly used when there is class label imbalance, and both classes hold the same importance in classification metric. In Equation 2, precision is the measure of the proportion of positive identifications that are actually correct, and recall is the measure of the proportion of actual positives that are correctly identified.

$$\text{F1-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{2}$$

To assess the DL model for regression problems, metrics like max variance, R-square, root mean squared error (RMSE), etc. are evaluated [198], [199], [200], [201], [202]. RMSE defined in equation 3, is a widely used metric for regression-based problems. In equation 3, y_i and $\hat{y}_i$ are the actual and predicted labels, and N is the total number of samples.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}. \tag{3}$$

While these metrics are widely used, they are essential for the performance comparison of different models' architecture and strategies deployed on the same dataset over the ES.

B. TRAINING LOSS

The process of training a DL model requires the optimization (typically minimization) of a specific loss function. The training loss is a metric that captures how well a DL model fits the training data by quantifying the loss between the predicted output and ground truth labels. Different metrics are selected based on the type of problem, *i.e.*, classification or regression. Some of the widely used loss functions to capture the learning of the DL model at the edge while training are mean absolute error [203], [204], [205], mean square error [206], [207], negative log-likelihood [208], [209], cross-entropy [210], [211], [212], Kullback-Leibler divergence [213], [214], [215] etc.

Cross-entropy, also called logarithmic loss, log loss, or logistic loss, is a widely accepted loss function for classification problems. In cross-entropy, the predicted class probability is compared to the actual class label. A loss is calculated

TABLE 5. Key performance metrics at all in-edge.

Performance Indicator	Metrics/ Descriptions
Use-case specific metrics	Accuracy, F1-score, Precision, Recall, R-square, Root mean squared error (RMSE), etc.
Training loss	Mean absolute error, Mean square error, Negative log-likelihood, Cross-entropy, etc.
Convergence rate	The number of iterations taken by a set of DL models (either having different model hyper-parameters or architecture) to converge on the same solution.
Latency	Computational latency and Communication latency
Communication cost	The amount of data (message size of each query) flowing to the deployed DL model at an ES from the end user. Communication cost is measured in kilobytes (KB) or megabytes (MB).
Privacy-preserving	Mutual information score (MIS), Structural similarity index measure (SSIM), Matthews correlation coefficient, etc.
Energy consumption	The amount of energy consumed by a set of DL models at an ES during the training and inference phase. Energy consumption is measured in watts and kilowatts.
Memory footprint	The amount of memory space (in RAM) required to host a set of DL models at ES during the training and inference phase. Model size or memory footprint is measured in megabytes (MB).
Combined metrics	Energy-precision ratio (EPR).
Robustness	KL Divergence
Transferability and lifelong learning	Transfer accuracy and Log expected empirical prediction (LEEP) score

that penalizes the DL model higher if the probability is very far from the actual value. The penalty itself is logarithmic, which yields a significant score for large differences close to 1 and small score for small differences tending to 0. The cross-entropy loss function is defined as

$$L(y, \hat{y}) = -\sum_{i=1}^{n} y_i \log(p_i)\ , \text{ for n classes}, \tag{4}$$

where p_i is the predicted class probability of the i^{th} class. Similarly, for regression problems, mean squared error (MSE) is the most commonly used loss function. The loss is the mean of the squared differences between true and predicted values across the dataset. MSE is defined as:

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N} (y - \hat{y}_i)^2 \,. \tag{5}$$

C. CONVERGENCE RATE

When training a DL model, we typically monitor its loss until it reaches some measure of convergence. We would expect the loss to decrease until any further updates to DL model parameters will not change the test dataset inference made by the DL model, known as the convergence of the DL model. The convergence rate is normally computed when using a distributed and decentralized architecture to train a DL model at the edge. One of the primary goals of the distributed/ decentralized DL model training at the edge is to speed up the convergence of DL models getting trained at multiple locations. Thus, DL models at different ESs need to collectively converge to a consensus that any further updates in the model will not change the estimate of the model for a given classification or regression problem [216]. Convergence rate, as a metric, defines the number of iterations one algorithm will take to converge to an optimum solution [217]. Thus, in a decentralized/ distributed architecture at all in-edge, the convergence rate as a metric becomes crucial because the different combinations of the architecture selected along with synchronization schemes (synchronous, asynchronous, etc.) have different convergence rates [218], [219], [220], [221].

D. LATENCY

When inferring from a model at the edge, both the computational latency and communication latency become critical key performance metrics. Computational latency provides an estimate of the time that the DL model will require to process a query input and infer on the same [222], [223], [224]. Whereas communication latency provides an estimate of the time from when a query is sent from the origin server until the result is returned [175], [225], [226], [227]. For mission-critical cases [228], DL models with low computational and communication latency are more favored. This metric becomes critical because one of the reasons to move from cloud to all in-edge is to reduce the latency incurred during the DL inference phase. The measuring unit of latency can range from milliseconds to seconds based on the latency requirement from DL-based applications.

E. COMMUNICATION COST

When a DL model is deployed for inference on an ES, many requests by the end user(s) are raised to get inference from the DL model. The volume of data, *e.g.,* kilobytes (KB) or megabytes (MB), transmitted from the end user(s) has the potential to create congestion at the ES. The communication cost metric evaluates the amount of data (message size of each query) flowing to the ES from the end user [56], [229]. It also takes into consideration the inference data, which is reverted to the end user. Active monitoring of the communication cost is important to prevent potential congestion points [230], [231], [232]. In typical cases, measuring the unit of communication cost is kept in KB or MB based on how much data is required to make an inference.

F. PRIVACY PRESERVING

Privacy-preserving metrics provide a means to quantify the level of user privacy offered by a DL model using privacy-preserving technologies [233]. We can assess the ability of a model to retain data privacy during the training and inference phases. In both phases, there are two types of data leakage: direct and indirect. Direct leakage at the training phase occurs when an external party gains access to non-encrypted training data sent to a centralized ES. In addition, direct leakage can also occur when in a decentralized/distributed setting when an external party gains access to activations or gradients that are sent from one edge server to another during training. Similarly, direct leakage at the inference phase occurs when an external party gains access to non-encrypted client data sent to an ES hosting the DL model. Indirect leakage at the training phase occurs when an external party gains access to DL model parameters which can indirectly provide information regarding training data. Indirect leakage from the inference phase comprises results provided by the DL model, which can leak sensitive information regarding the data DL model, is trained upon, i.e., membership inference attack and model inversion attack. Well-established encryption algorithms manage direct leakage from non-encrypted data at the training and inference phase, like DES, 3DES, AES, RSA, and blowfish, which don't require evaluation [234].

During the training phase, we can use the mutual information score (MIS) to measure the level of direct leakage (activation or gradients being sent from one edge server to another) or indirect leakage (access to DL model parameters) [121].

The mutual information score (represented as I, in equation 6) measures how much information a random variable X (*e.g.*, smashed data/ model parameters) can reveal about another random variable Y (*e.g.*, non-encrypted training data/ another set of model parameters). For X and Y with joint distribution of $p(x, y)$, it is defined as follows:

$$I(X, Y) = \sum_{x \in X, y \in Y} p(x, y) log \frac{p(x, y)}{p(x)p(y)}. \tag{6}$$

This metric ranges from 0 to 1, where 0 implies the raw data are independent of the intermediary activation vector or model parameters differ from another set of model parameters.

During the inference phase, two attacks (the model inversion attack and the memberships inference attack) can lead to indirect leakage. A model inversion attack allows an adversary to recover the confidential dataset utilized for training a supervised neural network. In an image-based model to evaluate model inversion, the structural similarity index measure (SSIM) is used to evaluate the reconstruction accuracy [235]. The magnitude of the deformation field resulting from non-linear registration of the original and reconstructed images is used to evaluate the reconstruction accuracy. The structural similarity index measure between two images x and y of common size $N \times N$ is:

$$\mathrm{SSIM}(x, y) = \frac{\left(2\mu_x\mu_y + c_1\right)\left(2\sigma_{xy} + c_2\right)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)}, \tag{7}$$

where

1) μ_x the average of x,
2) μ_y the average of y,
3) σ_x^2 the variance of x,
4) σ_y^2 the variance of y,
5) σ_{xy} the covariance of x and y,
6) $c_1 = (k_1L)^2, c_2 = (k_2L)^2$ two variables to stabilize the division with weak denominator,
7) L the dynamic range of the pixel-values (typically this is $2^{\#\text{ bits per pixel}} - 1$), and
8) $k_1 = 0.01$ and $k_2 = 0.03$ by default.

A membership inference attack, in contrast, does not recover the training data but allows an adversary to query a deployed DL model to infer whether or not a particular example was contained in the model's training dataset. An adversary in this approach trains another DL model to infer whether a specific example was present in the training dataset. Accuracy as a metric is utilized to evaluate the quality of the adversary's DL model. One of the current metrics proposed to measure membership inference is the ratio of the true-positive rate to false-positive rates. This metric provides more strict measures to make the DL model provide a guarantee that in an ideal scenario, none of the positive cases should be incorrectly identified. This metric becomes different from AUC- ROC curve as TPR is only reported for fixed low FPR (*e.g.*, 0.001% or 0.1%) [236].

G. ENERGY CONSUMPTION

There is a wide range of available DL models, and their individual energy requirements for computation can vary significantly. For some resource-constrained environments, it becomes infeasible to host models with a larger energy footprint [237]. The energy requirements of different models should be evaluated for their training and inference phase at the all in-edge level [238], [239], [240]. Power consumption (watts and kilowatts units) as measurement can be utilized to determine energy consumption [241]. Clearly, this metric is particularly relevant to an ES when hosting all the parts of a deep learning model.

H. MEMORY FOOTPRINT/MODEL SIZE

For an ES with limited computational resources, it can be challenging to host a DL model with a huge number of parameters. The larger the DL model the more parameters it will have, and consequently the more memory space (in RAM) required to host the model. Model size or memory footprint is computed having 'MB' as their unit of measurement [242], [243], [244], [245], [246]. For a specific image classification problem, if MobileNet V2 with 3.54 million parameters is selected, it will have 14 MB as model size whereas if InceptionV4 with 42.74 million parameters is selected for the same problem, it will have a 163 MB model size requirement [241].

I. COMBINED METRICS

As all in-edge needs to satisfy multiple constraints (*i.e.*, energy, quality of DL model, latency etc.), it becomes more important to introduce hybrid metrics that combine multiple metrics. For example, the energy-precision ratio (EPR) [247], provides a way to combine the classification error with the energy consumed per sample. In equation 8, energy-precision ratio (EPR) is defined as:

$$\text{EPR} = \text{Error}^{\alpha} \times \text{EPI}, \tag{8}$$

where Error is the classification error, α is the adjustment parameter, and EPI is the energy consumption per sample.

J. ROBUSTNESS

Adversarial examples can manipulate DL models, and negatively affect the models' performance or lead to misclassification. Thus the models need to be either robust to these examples by default or integrate various defence techniques to strengthen their robustness properties. The robustness of a model is defined as the insensitivity of the model to small perturbations made to any plausible input. Moreover, the robustness can be defined as the reciprocal of the KL Divergence:

$$\psi(x) = \frac{1}{\max\limits_{\delta \in set} D_{\text{KL}}(\hat{y}, \hat{y'})}, \tag{9}$$

where D_{KL} is the KL Divergence, $\hat{y}$ and $\hat{y'}$ are the predictions for a sample x and $x + \delta$, respectively [248]. Another simple way to measure the robustness can be the difference in the accuracy with and without the adversarial examples. The defence techniques for robustness include PixelDP, which is a certified defence for norm-bounded adversarial samples [249], adversarial training and ensemble learning.

K. TRANSFERABILITY AND LIFELONG LEARNING

The ability to reuse previously learned information for a related task indicates the transferability of a model. Thus, there is no need to train the model from scratch for a new task if the model is transferable from another related domain [250]. The transferability can be measured by using transfer accuracy and LEEP score ($T(\theta, D)$) [251].

$$T(\theta, D) = \frac{1}{n}\sum_{i=1}^{n}(\sum_{z \in Z} \hat{P}(y_i|z), \theta(x_i)_z), \tag{10}$$

where x_i is a sample, θ is the source model, D is the target dataset, $z \in Z$ is the label in the label set Z of the source task, and $\hat{P}(y_i|z)$ is the conditional distribution of the predicted labels given an original label.

The environment can change over time, and the model needs to adjust accordingly to capture the changes. Thus, models capable of lifelong learning are preferable. In lifelong learning, the model retains the previously gained knowledge and also keeps learning new information with time. Overall, transferability and lifelong learning capability make the DL models data and computation efficient.

V. OPEN CHALLENGES AND FUTURE DIRECTION

Thus far, we have discussed DL architectures, technologies, adaption techniques, and the key performance indicators required to facilitate DL to all in-edge. In this section, we now articulate the key open challenges and future research directions in the area of DL at all in-edge.

A. CHALLENGES WITH RESOURCE-CONSTRAINED EDGE SERVERS

It's necessary to know the configuration of ESs before starting the training and deployment of the DL model at ESs. This section discusses challenges that arise from heterogeneous ESs provided by different edge infrastructure providers (*e.g.*, Motorola Solutions, Hikvision, ADT) and associated future directions of research.

1) MEMORY EFFICIENCY

There are significant challenges to facilitating both the training and inference of DL models on ESs due to the limited resources and heterogeneous configuration of different ESs. DL models can vary significantly in their overall size. For example, inception-v3 has a size of 91 MB [252], while vgg-19 has a size of 548 MB [253]). Thus, based on the selected DL model (assuming it to be vgg-19) and enabling technology (assuming it to be federated learning), it can become impossible for some ES to participate in DL model training due to insufficient memory (if memory is less than 548 MB). The lack of availability of certain ESs can negatively impact the DL model convergence rate (a small number of available ESs for distributed training will mean a slower convergence rate). Also, due to the fairly large size, some DL models will be limited to being deployed for inference at a small number of ESs. In the future, explore the direction of utilization of heterogeneous ESs by answering: How can we train a DL model at ESs where some models can train fairly large models due to extensive memory, and some can partially train those models? How can we design DL models to facilitate training across heterogeneous ESs?

2) ENERGY REQUIREMENT

As ESs in remote locations can be battery-powered, minimizing energy consumption is a critical ongoing challenge. One way to achieve it is by limiting the computation required in the training and inference phase, which inherently lowers the energy requirement. Another important avenue of research is to investigate the performance of battery-operated ESs when different DL models are trained and deployed. While chipset designers continuously strive to reduce the energy requirements of their products (GPUs, TPUs, etc.). The same understanding of the interaction of the rest of ES composition (computing chipset, storage drives, batteries, etc. are required) to find a fair trade-off between battery management and compute resources is required.

B. QUALITY OF SERVICE (QoS) ATTRIBUTES FOR DL MODEL AT AN ALL IN-EDGE LEVEL

In order to be competitive with a centralized cloud model, the all in-edge model needs to provide quality of service guarantees. This section discusses the ''DL model at all in-edge guarantees'' to build a complete all in-edge framework for DL.

1) LOW LATENCY

Low latency is the first attribute that needs to be fulfilled at an all in-edge level. Low latency can be achieved by providing faster communication during model training and a quicker inference response from a deployed DL model. Due to the closer proximity of a deployed DL model to the end users, reduced latency has been observed in edge-based models compared to the traditional cloud-based models. For real-world applications, DL applications like image segmentation, object detection, etc., require very low latency. Using edge-based DL models, academic and industrial researchers actively seek ways to reduce latency [254], [255], [256], [257], [258]. Although progress has been made in this area, the current state-of-art still results in significant latency, specifically when dealing with high dimensional input data (*e.g.*, image, time series). For example, a constrained model architecture can process between 5 to 15 frames per second (fps) with an image resolution set to 1920 × 1080 [259]. However, processing 5 to 15 frames per second is relatively lower than the fps at which videos are captured (typically 24 fps higher). Processing a higher number of frames will result in delayed latency at the inference stage; This remains an open research problem and as such new techniques are required to deal with high dimensional data.

Similarly, model compression provides approaches to reduce latency by enabling larger DL models to be deployed at an ES. This reduces the computation required (as the DL model is quantized and compressed). However, DL networks have continued to grow in size (leading to a corresponding increase in the number of parameters). This necessitates further research on providing more powerful compression techniques for DL networks.

2) HETEROGENEOUS DATA DISTRIBUTION AND ASYNCHRONOUS EDGE SERVER PARTICIPATION

The second attribute required by the DL model at the all in-edge level is its ability to be trainable at ESs with heterogeneous data distribution. Heterogeneous data distribution is caused by the non-Identical and Independent Distribution (non-IID) of the data among the multiple ESs, which leads to severe statistical heterogeneity challenges when training a DL model. For example, one extreme case is when an end server only has data from a particular class. Usually, DL algorithms trained in a distributed environment with multiple ESs with an overall non-IID data distribution will perform poorly [260]. This opens an interesting future direction of research. Adaptive optimization is one approach that can be used to improve the convergence speed of a DL model and can effectively mitigate the concerns of non-IID data distribution. For example, [261] proposed adapting FedAvg to use a distributed form of Adam optimization to implement adaptive federated learning, which converges to a target accuracy in 6× fewer rounds than compressed FedAvg. In the future, exploring momentum, adaptive optimization, learning rates, and other hyperparameters is a worthwhile research direction in the context of a non-IID distribution. In addition, the participation of the ESs can be inconsistent due to communication or computational reasons. This results in either a slowdown in the convergence rate or an inability to converge at all [262], thus placing more emphasis on asynchronous ES participation in the training phase. Future research into the robustness of training models in such scenarios is also warranted.

C. PRIVACY AND SECURITY CONCERNS

Despite the rapid development of privacy-preserving DL [263] and security mitigation techniques [264] in recent years, there are still open research challenges that need to be addressed. This section discusses potential open research problems and future directions regarding privacy and security concerns impacting DL model development and deployment at all in-edge.

1) PRIVACY-PRESERVATION

Providing adequate privacy preservation for DL applications is an area with open research challenges. To preserve the privacy of the client's data at the ES, different enabling technologies are utilized with or without cryptographic techniques, perturbation techniques, and anonymization techniques [265], [266]. While these techniques provide a means of better safeguarding client data, they struggle to maintain the original level of model performance simultaneously. For example, the inclusion of these techniques can not only negatively impact the predictive performance of a model (accuracy, F1-score, etc) [103], [267], [268] but can also significantly lengthen the training [99], [269], and inference [270], [271]) time of a model. Therefore, there are opportunities in this area to preserve privacy while mitigating the negative consequences outlined above.

2) SECURITY

The ESs need active participation while enabling DL at the all in-edge level. However, due to hardware constraints (*e.g.*, low computational capability) and software heterogeneities of the ESs, this also represents an increase in the attack surface. Moreover, various attacks such as Distributed Denial-of-service(DDoS) targeting network/virtualization infrastructure, side-channel attacks targetting user data/privacy, malware injection targeting ES/devices, authentication and authorization attacks targeting ES/devices and virtualization infrastructure are possible for all in-edge computing system [272]. However, finding efficient and suitable countermeasures for these attacks is challenging due to constantly evolving attackers' tactics, techniques, and procedures [273]. Besides, DL approaches such as federated learning and split learning for edge intelligence suffer from adversarial attacks on the federated models to modify their behavior and extract/reconstruct original data [274]. To that end, novel techniques are required to identify security attacks/breaches and mitigate such attacks in the future.

D. FRAMEWORK AND ARCHITECTURAL CHANGES TO FACILITATE DL MODELS AT ALL IN-EDGE LEVEL

The convergence of DL at the all in-edge level is a relatively new paradigm, with concerns about effective resource utilization, management, and interoperability amongst heterogeneous ESs, requiring new frameworks and architectural changes. This section will discuss promising directions that can help mitigate the concerns at the convergence.

1) MICROSERVICES

As computing is getting pushed away from being cloud-based to edge-based, architectures to facilitate DL model training and deployment are also shifting from monolithic entities to graphs of loosely-coupled microservices [275]. Microservices provide a promising way of modularizing DL-based applications at the process level. For example, a single DL application can be decomposed into a non-overlapping atomic set of services in a microservice architecture. However, sometimes one DL model inference can depend on another DL model inference. At the same time, another DL model may require different computing languages (*e.g.*, python, R), language dependencies (*e.g.*, PyTorch, TensorFlow), and software dependencies (*e.g.*, pycharm, GitLab). Microservices architectures provide a means for those DL models with different requirements to communicate effectively. Currently, the introduction of microservices for deploying and training at the edge is at a very early stage [276]. The research opportunity exists to build a robust microservice framework that can handle the deployment and management of the DL model. Another opportunity lies in migrating microservices-based DL applications from development to production with minimal downtime.

2) MANAGEMENT OF DL-BASED APPLICATIONS AT ESs

The confluence of DL models deployed at ESs and the emergence of smart cities has led to a new interesting research area of DL-assisted smart cities. With many DL models deployed at ESs in smart cities, it will become challenging to predict the future requirement of resources for DL computation accurately. Real-time optimization will be required amongst ESs to accommodate heterogeneous computation and communication adaptively. As a result, better resource orchestrators (online ES management applications) will be required at the edge to facilitate the potentially large number of requests that will be generated within an ecosystem of smart cities. Also, with every government taking steps toward smart cities, these orchestrations will be dispersed across different geolocations and regions, thus providing an opportunity for collaboration between individual orchestrators. A flexible coordination mechanism between orchestrators situated adjacent to each other will be required, which can also preserve citizens' privacy. An emerging research direction is utilizing AI to tackle the design complexity of interconnected smart cities; One of the ways to achieve it is by using deep reinforcement learning (DRL) [277]. A distributed DRL-based scheme can provide an efficient way to solve the data-driven interference mitigation and resource allocation problem. It also opens up new research opportunities on the need to develop a uniform API interface for ubiquitous heterogeneous ESs to ease the deployment of orchestrators. Due to the highly dynamic nature of this environment (any ES can go offline and come back into service), an important and related research direction is the design of efficient service discovery protocols. Service discovery protocols will provide necessary information to companion ESs regarding what can be expected from DL-based applications deployed at that ES.

3) DESIGNING APPLICATION FRAMEWORK TO FACILITATE DL AT THE ALL IN-EDGE LEVEL

All in-edge paradigm requires new ways of designing applications. In Section III-A, we presented different architectures capable of pushing AI to the ES with varying application requirements. With the enabling technologies explained in Section III-B) and model adaption techniques described in Section III-C, developing DL applications becomes progressively more complex. The aforementioned microservices-based architecture is another exciting area of research in the provisioning of DL-based applications at ESs [278]. Although other research provided the framework for designing DL-based applications by utilizing ESs, they all remain confined to the problem they tried to resolve. For example, in [279] provided a framework for the self-learning DL model, in which authors proposed a GAN-based synthesis of the traffic images. The proposed framework remains applicable only for video-based scenarios. Similarly, the work in [280] provides a framework that was restricted to work for web traffic anomaly detection. Likewise, other research [281], [282] has its niche, and the proposed framework is restricted to solving the specific problem type. To the author's knowledge, Open EI [283] is the only framework that provides a generic approach to facilitate the development of applications for a wide range of problem domains (computer vision, natural language processing, etc.). Still, this framework lacks the components of hardware (choices in the selection of hardware accelerators that can help in faster DNN computation [284], [285], [286], [287]) and the deployment of the DL-based services (how to distribute load and develop a global model across the ES III-B). Therefore, there is a need to find a robust framework that can facilitate the easy development and deployment of complex DL-based applications at the all in-edge level by providing guarantees from DL-based applications (as mentioned in Section V-B) while adhering to infrastructural constraints of the ES resources (as discussed in Section V-A) alongside mitigating the privacy and security concerns (as described in Section V-C).

VI. CONCLUSION

This paper reviewed the current states to facilitate the training and inference of DL models on a fine mesh of ESs (referred to as all in-edge level). The behavior of centralized, decentralized, and distributed architecture were discussed from the ES's perspective to find a trade-off between simplicity (by centralized architecture) or achieving reliability (by utilizing a decentralized and distributed architecture) for DL models deployed at the all in-edge level. Technologies facilitating the DL training and deployment across ESs were described, which leverage the layer structure of the DL models and closer proximity to the origin of the data. Federated learning and split learning as enabling technologies were more effective than others as they provided enhanced privacy while training and providing inference from the DL model. Model adaption techniques were found to be necessary at all in-edge, providing benefits of minimizing the energy requirement, lowering communication message size, and decreasing the memory footprint. In addition to general performance indicators, this paper identified and put forward additional key performance indicators, measured in silos in several works but not considered to be evaluated simultaneously. Many research directions remain open regarding optimizing memory and energy of resource-constrained ESs for facilitating DL at ESs while preserving the privacy of the user's data, incorporating advancements in cybersecurity to diminish security concerns, and lastly, close collaboration with networking technologies (such as network functions virtualization). With new technological innovations, shifts in DL-based application design, networking technologies improvements, and ESs hardware advances, many of the previously mentioned challenges will be mitigated. This will bring new challenges and opportunities for further innovation.

REFERENCES

[1] *IMT Traffic Estimates for the Years 2020 to 2030*, document ITU-R M.2370-0, ITU-R, 2020, Accessed: Oct. 11, 2022. [Online]. Available: https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2370-2015-PDF-E.pdf

[2] K. Aggarwal, M. M. Mijwil, A. H. Al-Mistarehi, S. Alomari, M. Gök, A. M. Z. Alaabdin, and S. H. Abdulrhman, "Has the future started? The current growth of artificial intelligence, machine learning, and deep learning," *Iraqi J. Comput. Sci. Math.*, vol. 3, no. 1, pp. 115–123, 2022.

[3] H. Lv, S. Shi, and D. Gursoy, "A look back and a leap forward: A review and synthesis of big data and artificial intelligence literature in hospitality and tourism," *J. Hospitality Marketing Manag.*, vol. 31, no. 2, pp. 145–175, Feb. 2022.

[4] R. Dale, "GPT-3: What's it good for?" *Natural Lang. Eng.*, vol. 27, no. 1, pp. 113–118, 2021.

[5] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz, "ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1–11.

[6] M. K. Patrick, A. F. Adekoya, A. A. Mighty, and B. Y. Edward, "Capsule networks—A survey," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 1, pp. 1295–1310, 2022.

[7] B. Wu, "Hierarchical macro strategy model for MOBA game AI in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 1206–1213.

[8] P. P. Ray, D. Dash, and D. De, "Edge computing for Internet of Things: A survey, e-healthcare case study and future direction," *J. Netw. Comput. Appl.*, vol. 140, pp. 1–22, Aug. 2019.

[9] N. A. Sulieman, L. R. Celsi, W. Li, A. Zomaya, and M. Villari, "Edge-oriented computing: A survey on research and use cases," *Energies*, vol. 15, no. 2, p. 452, Jan. 2022.

[10] A. Alsalemi, Y. Himeur, F. Bensaali, and A. Amira, "An innovative edge-based internet of energy solution for promoting energy saving in buildings," *Sustain. Cities Soc.*, vol. 78, Mar. 2022, Art. no. 103571.

[11] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep. 2019.

[12] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1135–1143.

[13] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[14] M. M. Sadeeq, N. M. Abdulkareem, S. R. M. Zeebaree, D. M. Ahmed, A. S. Sami, and R. R. Zebari, "IoT and cloud computing issues, challenges and opportunities: A review," *Qubahan Academic J.*, vol. 1, no. 2, pp. 1–7, Mar. 2021.

[15] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020.

[16] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[17] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Jul. 2019.

[18] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proc. IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

[19] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Comput. Surveys*, vol. 54, no. 8, pp. 1–37, Nov. 2022.

[20] A. Salem and O. Moselhi, "AI-based cloud computing application for smart earthmoving operations," *Can. J. Civil Eng.*, vol. 48, no. 3, pp. 312–327, Mar. 2021.

[21] Z. Chen, G. Xu, V. Mahalingam, L. Ge, J. Nguyen, W. Yu, and C. Lu, "A cloud computing based network monitoring and threat detection system for critical infrastructures," *Big Data Res.*, vol. 3, pp. 10–23, Apr. 2016.

[22] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the Internet of Things: A survey," *ACM Trans. Internet Technol.*, vol. 19, pp. 1–41, Apr. 2019.

[23] M. A. Dantas, P. E. Bogoni, and P. J. D. F. Filho, "An application study case tradeoff between throughput and latency on fog-cloud cooperation," *Int. J. Netw. Virtual Organisations*, vol. 23, no. 3, pp. 247–260, 2020.

[24] Z. Ali, S. Khaf, Z. H. Abbas, G. Abbas, F. Muhammad, and S. Kim, "A deep learning approach for mobility-aware and energy-efficient resource allocation in MEC," *IEEE Access*, vol. 8, pp. 179530–179546, 2020.

[25] A. Barate, G. Haus, L. A. Ludovico, E. Pagani, and N. Scarabottolo, "5G technology for augmented and virtual reality in education," in *Proc. Int. Conf. Educ. New Develop.*, 2019, pp. 512–516.

[26] V. Kupriyanovsky, A. Klimov, I. Sokolov, and O. Pokusaev, "EU digital transport corridors-5G, platooning, ITS and MaaS," *Int. J. Open Inf. Technol.*, vol. 7, no. 8, pp. 70–86, 2019.

[27] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for D2D-enabled partial computation offloading in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4472–4486, Apr. 2020.

[28] W. Li and M. Liewig, "A survey of AI accelerators for edge environment," in *Proc. World Conf. Inf. Syst. Technol.* Cham, Switzerland: Springer, 2020, pp. 35–44.

[29] G. E. Sánchez-Martínez and M. Munizaga, "Workshop 5 report: Harnessing big data," *Res. Transp. Econ.*, vol. 59, pp. 236–241, Nov. 2016.

[30] S. Milz, G. Arbeiter, C. Witt, B. Abdallah, and S. Yogamani, "Visual SLAM for automated driving: Exploring the applications of deep learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 247–257.

[31] J. Barthélemy, N. Verstaevel, H. Forehead, and P. Perez, "Edge-computing video analytics for real-time traffic monitoring in a smart city," *Sensors*, vol. 19, no. 9, p. 2048, May 2019.

[32] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 695–708, Jan. 2019.

[33] N. Hassan, S. Gillani, E. Ahmed, I. Ibrar, and M. Imran, "The role of edge computing in Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 11, pp. 110–115, Nov. 2018.

[34] O. Oderhohwo, H. Mohammed, T. Odetola, T. N. Guo, S. Hasan, and F. Dogbe, "An edge intelligence framework for resource constrained community area network," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2020, pp. 97–100.

[35] B. Charyyev, E. Arslan, and M. H. Gunes, "Latency comparison of cloud datacenters and edge servers," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[36] Y. Huang, X. Qiao, W. Lai, S. Dustdar, J. Zhang, and J. Li, "Enabling DNN acceleration with data and model parallelization over ubiquitous end devices," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 15053–15065, Aug. 2022.

[37] K. R. R. Devi, R. Murugesan, and R. M. Chozhan, "Cloud-based CVD identification for periodontal disease," in *Machine Learning and Autonomous Systems* (Smart Innovation, Systems and Technologies). Berlin, Germany: Springer, 2022, pp. 591–607.

[38] X. Kong, K. Wang, S. Wang, X. Wang, X. Jiang, Y. Guo, G. Shen, X. Chen, and Q. Ni, "Real-time mask identification for COVID-19: An edge-computing-based deep learning framework," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15929–15938, Nov. 2021.

[39] L. Kong, T. Lin, A. Koloskova, M. Jaggi, and S. Stich, "Consensus control for decentralized deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 5686–5696.

[40] J. K. Kim, Q. Ho, S. Lee, X. Zheng, W. Dai, G. A. Gibson, and E. P. Xing, "Strads: A distributed framework for scheduled model parallel machine learning," in *Proc. 11th Eur. Conf. Comput. Syst.*, 2016, pp. 1–16.

[41] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, A. Phanishayee, and M. Zaharia, "Efficient large-scale language model training on GPU clusters using megatron-LM," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2021, pp. 1–15.

[42] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, "RACE: Large-scale reading comprehension dataset from examinations," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 785–794. [Online]. Available: https://aclanthology.org/D17-1082

[43] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *Proc. 14th USENIX Symp. Networked Syst. Design Implement.*, 2017, pp. 629–647.

[44] N. Loizou and P. Richtarik, "A new perspective on randomized gossip algorithms," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Dec. 2016, pp. 440–444.

[45] M. Blot, D. Picard, N. Thome, and M. Cord, "Distributed optimization for deep learning with gossip exchange," *Neurocomputing*, vol. 330, pp. 287–296, Feb. 2019.

[46] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging SGD," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.

[47] R. Sajina, N. Tankovic, and D. Etinger, "Decentralized trustless gossip training of deep neural networks," in *Proc. 43rd Int. Conv. Inf., Commun. Electron. Technol. (MIPRO)*, Sep. 2020, pp. 1080–1084.

[48] H. Oguni and K. Shudo, "Communication scheduling for gossip SGD in a wide area network," *IEEE Access*, vol. 9, pp. 77873–77881, 2021.

[49] R. Han, S. Li, X. Wang, C. H. Liu, G. Xin, and L. Y. Chen, "Accelerating gossip-based deep learning in heterogeneous edge computing platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1591–1602, Jul. 2021.

[50] S. Han and W. J. Dally, "Bandwidth-efficient deep learning," in *Proc. 55th Annu. Design Autom. Conf.*, Jun. 2018, pp. 1–6.

[51] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 7652–7662.

[52] Y. Du, S. Yang, and K. Huang, "High-dimensional stochastic gradient quantization for communication-efficient edge learning," *IEEE Trans. Signal Process.*, vol. 68, pp. 2128–2142, 2020.

[53] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "SIGNSGD: Compressed optimisation for non-convex problems," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 560–569.

[54] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. Interspeech*, Sep. 2015, pp. 1–5.

[55] Z. Tao and Q. Li, "eSGD: Communication efficient distributed deep learning on the edge," in *Proc. USENIX Workshop Hot Topics Edge Comput.*, 2018, pp. 1–6.

[56] S. Shi, Q. Wang, X. Chu, B. Li, Y. Qin, R. Liu, and X. Zhao, "Communication-efficient distributed deep learning with merged gradient sparsification on GPUs," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 406–415.

[57] N. Negi, O. Jelassi, H. Chaouchi, and S. Clemencon, "Distributed online data anomaly detection for connected vehicles," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, Feb. 2020, pp. 494–500.

[58] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, and S. Chintala, "PyTorch distributed: Experiences on accelerating data parallel training," 2020, *arXiv:2006.15704*.

[59] M. Szabó, "Distributed machine learning using data parallelism on mobile platform," *J. Mobile Multimedia*, vol. 16, pp. 317–334, Sep. 2020.

[60] 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.

[61] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. Nguyen, and C. S. Hong, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 88–93, Oct. 2020.

[62] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 1146–1159, Nov. 2019.

[63] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, B. McMahan, and T. Van Overveldt, "Towards federated learning at scale: System design," in *Proc. Mach. Learn. Syst.*, vol. 1, 2019, pp. 374–388.

[64] 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.*, 2020, pp. 2021–2031.

[65] G. Long, M. Xie, T. Shen, T. Zhou, X. Wang, and J. Jiang, "Multi-center federated learning: Clients clustering for better personalization," *World Wide Web*, pp. 1–20, Jun. 2022.

[66] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, early access, Nov. 2, 2021, doi: 10.1109/TKDE.2021.3124599.

[67] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Gener. Comput. Syst.*, vol. 115, pp. 619–640, Feb. 2021.

[68] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, Mar. 2021, Art. no. 106775.

[69] L. Lyu, H. Yu, J. Zhao, and Q. Yang, "Threats to federated learning," in *Federated Learning*. Berlin, Germany: Springer, 2020, pp. 3–16.

[70] Q. Zhang, B. Gu, C. Deng, S. Gu, L. Bo, J. Pei, and H. Huang, "AsySQN: Faster vertical federated learning algorithms with better computation resource utilization," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 3917–3927.

[71] Z. Zhou, Y. Tian, and C. Peng, "Privacy-preserving federated learning framework with general aggregation and multiparty entity matching," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–14, Jun. 2021.

[72] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–36, 2021.

[73] D. Stripelis, H. Saleem, T. Ghai, N. J. Dhinagar, U. Gupta, C. Anastasiou, G. V. Steeg, S. Ravi, M. Naveed, P. M. Thompson, and J. L. Ambite, "Secure neuroimaging analysis using federated learning with homomorphic encryption," in *Proc. 17th Int. Symp. Med. Inf. Process. Anal.*, Dec. 2021, pp. 351–359.

[74] A. Madi, O. Stan, A. Mayoue, A. Grivet-Sebert, C. Gouy-Pailler, and R. Sirdey, "A secure federated learning framework using homomorphic encryption and verifiable computing," in *Proc. Reconciling Data Anal., Automat., Privacy, Secur., Big Data Challenge (RDAAPS)*, 2021, pp. 1–8.

[75] L. Zhang, J. Xu, P. Vijayakumar, P. K. Sharma, and U. Ghosh, "Homomorphic encryption-based privacy-preserving federated learning in IoT-enabled healthcare system," *IEEE Trans. Netw. Sci. Eng.*, early access, Jun. 30, 2022, doi: 10.1109/TNSE.2022.3185327.

[76] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, p. 94, Apr. 2021.

[77] A. Mondal, Y. More, R. H. Rooparaghunath, and D. Gupta, "Poster: FLATEE: Federated learning across trusted execution environments," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Sep. 2021, pp. 707–709.

[78] W. Mou, C. Fu, Y. Lei, and C. Hu, "A verifiable federated learning scheme based on secure multi-party computation," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl.* Cham, Switzerland: Springer, 2021, pp. 198–209.

[79] E. Sotthiwat, L. Zhen, Z. Li, and C. Zhang, "Partially encrypted multi-party computation for federated learning," in *Proc. IEEE/ACM 21st Int. Symp. Cluster, Cloud Internet Comput. (CCGrid)*, May 2021, pp. 828–835.

[80] A. Girgis, D. Data, S. Diggavi, P. Kairouz, and A. T. Suresh, "Shuffled model of differential privacy in federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 2521–2529.

[81] Z. Zhang, L. Zhang, Q. Li, K. Wang, N. He, and T. Gao, "Privacy-enhanced momentum federated learning via differential privacy and chaotic system in industrial cyber–physical systems," *ISA Trans.*, vol. 128, pp. 17–31, Sep. 2022.

[82] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "LDP-Fed: Federated learning with local differential privacy," in *Proc. 3rd ACM Int. Workshop Edge Syst., Anal. Netw.*, 2020, pp. 61–66.

[83] Y. Zhao, J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, and K. Y. Lam, "Local differential privacy-based federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8836–8853, Nov. 2020.

[84] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, "FedAT: A high-performance and communication-efficient federated learning system with asynchronous tiers," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2021, pp. 1–16.

[85] Q. Zhang, B. Gu, C. Deng, and H. Huang, "Secure bilevel asynchronous vertical federated learning with backward updating," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 12, pp. 10896–10904.

[86] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3654–3672, Dec. 2021.

[87] Y. Wan, Y. Qu, L. Gao, and Y. Xiang, "Privacy-preserving blockchain-enabled federated learning for B5G-driven edge computing," *Comput. Netw.*, vol. 204, Feb. 2022, Art. no. 108671.

[88] S. K. Lo, Q. Lu, H.-Y. Paik, and L. Zhu, "FLRA: A reference architecture for federated learning systems," in *Proc. Eur. Conf. Softw. Archit.* Cham, Switzerland: Springer, 2021, pp. 83–98.

[89] J.-H. Chen, M.-R. Chen, G.-Q. Zeng, and J.-S. Weng, "BDFL: A Byzantine-fault-tolerance decentralized federated learning method for autonomous vehicle," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 8639–8652, Sep. 2021.

[90] K. Toyoda, J. Zhao, A. N. S. Zhang, and P. T. Mathiopoulos, "Blockchain-enabled federated learning with mechanism design," *IEEE Access*, vol. 8, pp. 219744–219756, 2020.

[91] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12806–12825, Aug. 2021.

[92] M. A. Rahman, M. S. Hossain, M. S. Islam, N. A. Alrajeh, and G. Muhammad, "Secure and provenance enhanced internet of health things framework: A blockchain managed federated learning approach," *IEEE Access*, vol. 8, pp. 205071–205087, 2020.

[93] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4298–4311, Apr. 2020.

[94] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus," *IEEE Netw.*, vol. 35, no. 1, pp. 234–241, Jan. 2021.

[95] L. Barbieri, S. Savazzi, M. Brambilla, and M. Nicoli, "Decentralized federated learning for extended sensing in 6G connected vehicles," *Veh. Commun.*, vol. 33, Jan. 2022, Art. no. 100396.

[96] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr, "SpreadGNN: Decentralized multi-task federated learning for graph neural networks on molecular data," in *Proc. 36th AAAI Conf. Artif. Intell.*, 2022, pp. 1–9.

[97] H. Xing, O. Simeone, and S. Bi, "Decentralized federated learning via SGD over wireless D2D networks," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, May 2020, pp. 1–5.

[98] T. Liu, P. Li, and Y. Gu, "Glint: Decentralized federated graph learning with traffic throttling and flow scheduling," in *Proc. IEEE/ACM 29th Int. Symp. Quality Service (IWQOS)*, Jun. 2021, pp. 1–10.

[99] C. Thapa, M. A. P. Chamikara, and S. A. Camtepe, "Advancements of federated learning towards privacy preservation: From federated learning to split learning," in *Federated Learning Systems* (Studies in Computational Intelligence). Cham, Switzerland: Springer, 2021, pp. 79–109.

[100] S. Abuadbba, K. Kim, M. Kim, C. Thapa, S. A. Camtepe, Y. Gao, H. Kim, and S. Nepal, "Can we use split learning on 1D CNN models for privacy preserving training?" in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 305–318.

[101] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," 2018, *arXiv:1812.00564*.

[102] C. Thapa, M. A. P. Chamikara, S. Camtepe, and L. Sun, "SplitFed: When federated learning meets split learning," 2020, *arXiv:2004.12088*.

[103] A. Pant, "Comparison of privacy-preserving distributed deep learning methods in healthcare," in *Proc. Annu. Conf. Med. Image Understand. Anal.*, vol. 12722. Oxford, U.K.: Springer, 2021, pp. 457–471.

[104] J. Geng, D. Li, and S. Wang, "ElasticPipe: An efficient and dynamic model-parallel solution to DNN training," in *Proc. 10th Workshop Sci. Cloud Comput.*, 2019, pp. 5–9.

[105] Y. Mao, S. Yi, Q. Li, J. Feng, F. Xu, and S. Zhong, "A privacy-preserving deep learning approach for face recognition with edge computing," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2018, pp. 1–6.

[106] A. Xu, Z. Huo, and H. Huang, "On the acceleration of deep learning model parallelism with staleness," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2088–2097.

[107] J. Yoon, Y. Byeon, J. Kim, and H. Lee, "EdgePipe: Tailoring pipeline parallelism with deep neural networks for volatile wireless edge devices," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 11633–11647, Jul. 2022.

[108] I. Hegedus, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *J. Parallel Distrib. Comput.*, vol. 148, pp. 109–124, Feb. 2021.

[109] M. A. Dinani, A. Holzer, H. Nguyen, M. A. Marsan, and G. Rizzo, "Gossip learning of personalized models for vehicle trajectory prediction," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Mar. 2021, pp. 1–7.

[110] S. Nikolaidis and I. Refanidis, "Using distributed ledger technology to democratize neural network training," *Appl. Intell.*, vol. 51, no. 11, pp. 8288–8304, 2021.

[111] L. Abrahamyan, Y. Chen, G. Bekoulis, and N. Deligiannis, "Learned gradient compression for distributed deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7330–7344, Dec. 2022.

[112] C.-Y. Chen, J. Ni, S. Lu, X. Cui, P. Y. Chen, X. Sun, N. Wang, S. Venkataramani, V. V. Srinivasan, W. Zhang, and K. Gopalakrishnan, "ScaleCom: Scalable sparsified gradient compression for communication-efficient distributed training," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 13551–13563.

[113] H. Li, A. Kadav, E. Kruus, and C. Ungureanu, "MALT: Distributed data-parallelism for existing ML applications," in *Proc. 10th Eur. Conf. Comput. Syst.*, Apr. 2015, pp. 1–16.

[114] R. Yu and P. Li, "Toward resource-efficient federated learning in mobile edge computing," *IEEE Netw.*, vol. 35, no. 1, pp. 148–155, Jan./Feb. 2021.

[115] J. H. Park, G. Yun, M. Y. Chang, N. T. Nguyen, S. Lee, J. Choi, S. H. Noh, and Y.-R. Choi, "Hetpipe: Enabling large DNN training on (Whimpy) heterogeneous GPU clusters through integration of pipelined model parallelism and data parallelism," in *Proc. USENIX Annu. Tech. Conf.*, 2020, pp. 307–321.

[116] Y. Li, Z. Zeng, J. Li, B. Yan, Y. Zhao, and J. Zhang, "Distributed model training based on data parallelism in edge computing-enabled elastic optical networks," *IEEE Commun. Lett.*, vol. 25, no. 4, pp. 1241–1244, Apr. 2021.

[117] A. Cheikh, S. Sordillo, A. Mastrandrea, F. Menichelli, G. Scotti, and M. Olivieri, "Klessydra-T: Designing vector coprocessors for multi-threaded edge-computing cores," *IEEE Micro*, vol. 41, no. 2, pp. 64–71, Mar. 2021.

[118] P. Chen, X. Du, Z. Lu, J. Wu, and P. C. K. Hung, "EVFL: An explainable vertical federated learning for data-oriented artificial intelligence systems," *J. Syst. Archit.*, vol. 126, May 2022, Art. no. 102474.

[119] X. Jin, P.-Y. Chen, C.-Y. Hsu, C.-M. Yu, and T. Chen, "CAFE: Catastrophic data leakage in vertical federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 994–1006.

[120] C. Thapa, P. C. M. Arachchige, S. Camtepe, and L. Sun, "SplitFED: When federated learning meets split learning," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 8, pp. 8485–8493.

[121] P. Joshi, C. Thapa, S. Camtepe, M. Hasanuzzaman, T. Scully, and H. Afli, "Performance and information leakage in splitfed learning and multi-head split learning in healthcare data and beyond," *Methods Protocols*, vol. 5, no. 4, p. 60, Jul. 2022.

[122] D. Y. Zhang, Z. Kou, and D. Wang, "FedSens: A federated learning approach for smart health sensing with class imbalance in resource constrained edge computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2021, pp. 1–10.

[123] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang, H. Yang, and W. J. Dally, "ESE: Efficient speech recognition engine with sparse LSTM on FPGA," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2017, pp. 75–84.

[124] Z. Li, Y. Gong, X. Ma, S. Liu, M. Sun, Z. Zhan, Z. Kong, G. Yuan, and Y. Wang, "SS-auto: A single-shot, automatic structured weight pruning framework of DNNs with ultra-high efficiency," 2020, *arXiv:2001.08839*.

[125] Y. Gong, Z. Zhan, Z. Li, W. Niu, X. Ma, W. Wang, B. Ren, C. Ding, X. Lin, X. Xu, and Y. Wang, "A privacy-preserving-oriented DNN pruning and mobile acceleration framework," in *Proc. Great Lakes Symp. VLSI*, Sep. 2020, pp. 119–124.

[126] D. Yang, W. Yu, H. Mu, and G. Yao, "Dynamic programming assisted quantization approaches for compressing normal and robust DNN models," in *Proc. 26th Asia South Pacific Design Autom. Conf.*, Jan. 2021, pp. 351–357.

[127] S. Huang, A. Ankit, P. Silveira, R. Antunes, S. R. Chalamalasetti, I. El Hajj, D. E. Kim, G. Aguiar, P. Bruel, S. Serebryakov, C. Xu, C. Li, P. Faraboschi, J. P. Strachan, D. Chen, K. Roy, W.-M. Hwu, and D. Milojicic, "Mixed precision quantization for ReRAM-based DNN inference accelerators," in *Proc. 26th Asia South Pacific Design Autom. Conf.*, Jan. 2021, pp. 372–377.

[128] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, pp. 1789–1819, Mar. 2021.

[129] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 5191–5198.

[130] H. Tsunashima, H. Kataoka, J. Yamato, Q. Chen, and S. Morishima, "Adversarial knowledge distillation for a compact generator," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 10636–10643.

[131] F. Yuan, L. Shou, J. Pei, W. Lin, M. Gong, Y. Fu, and D. Jiang, "Reinforced multi-teacher selection for knowledge distillation," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 16, pp. 14284–14291.

[132] K. Wang, Y. Liu, Q. Ma, and Q. Z. Sheng, "MulDE: Multi-teacher knowledge distillation for low-dimensional knowledge graph embeddings," in *Proc. Web Conf.*, Apr. 2021, pp. 1716–1726.

[133] Z. Hao, Y. Luo, Z. Wang, H. Hu, and J. An, "Model compression via collaborative data-free knowledge distillation for edge intelligence," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2021, pp. 1–6.

[134] F. M. Thoker and J. Gall, "Cross-modal knowledge distillation for action recognition," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 6–10.

[135] J. Sun, L. Zhang, Y. Zha, A. Gonzalez-Garcia, P. Zhang, W. Huang, and Y. Zhang, "Unsupervised cross-modal distillation for thermal infrared tracking," in *Proc. 29th ACM Int. Conf. Multimedia*, Oct. 2021, pp. 2262–2270.

[136] P. Passban, Y. Wu, M. Rezagholizadeh, and Q. Liu, "ALP-KD: Attention-based layer projection for knowledge distillation," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 15, pp. 13657–13665.

[137] H. Inaguma and T. Kawahara, "Alignment knowledge distillation for online streaming attention-based speech recognition," *IEEE/ACM Trans. Audio, Speech, Language Process.*, early access, Dec. 7, 2021, doi: 10.1109/TASLP.2021.3133217.

[138] C. You, N. Chen, and Y. Zou, "Contextualized attention-based knowledge transfer for spoken conversational question answering," 2021, *arXiv:2010.11066*.

[139] D. Chen, J.-P. Mei, Y. Zhang, C. Wang, Z. Wang, Y. Feng, and C. Chen, "Cross-layer distillation with semantic calibration," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 8, pp. 7028–7036.

[140] Y.-S. Chuang, S.-Y. Su, and Y.-N. Chen, "Lifelong language knowledge distillation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 2914–2924.

[141] A. Yao and D. Sun, "Knowledge transfer via dense cross-layer mutual-distillation," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 294–311.

[142] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Q-bert: Hessian based ultra low precision quantization of bert," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 8815–8821.

[143] Y. Boo, S. Shin, J. Choi, and W. Sung, "Stochastic precision ensemble: Self-knowledge distillation for quantized deep neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 8, pp. 6794–6802.

[144] Z. Meng, J. Li, Y. Zhao, and Y. Gong, "Conditional teacher-student learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 6445–6449.

[145] S. Jain, S. Hamidi-Rad, and F. Racape, "Low rank based end-to-end deep neural network compression," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2021, pp. 233–242.

[146] D. Papadimitriou and S. Jain, "Data-driven low-rank neural network compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 3547–3551.

[147] F. Patrona, I. Mademlis, and I. Pitas, "Self-supervised convolutional neural networks for fast gesture recognition in human–robot interaction," in *Proc. 10th Int. Conf. Inf. Autom. Sustainability (ICIAfS)*, Aug. 2021, pp. 88–93.

[148] K. Han, Y. Wang, C. Xu, C. Xu, E. Wu, and D. Tao, "Learning versatile convolution filters for efficient visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 7731–7746, Nov. 2022.

[149] S. Lee, H. Kim, B. Jeong, and J. Yoon, "A training method for low rank convolutional neural networks based on alternating tensor compose-decompose method," *Appl. Sci.*, vol. 11, no. 2, p. 643, Jan. 2021.

[150] F. Yang, W. Liu, J. Liu, C. Liu, Y. Mi, and H. Song, "Iterative low-rank approximation based on the redundancy of each network layer," in *Proc. SPIE*, vol. 11720, Jan. 2021, Art. no. 117202G.

[151] E. Russo, M. Palesi, S. Monteleone, D. Patti, A. Mineo, G. Ascia, and V. Catania, "DNN model compression for IoT domain-specific hardware accelerators," *IEEE Internet Things J.*, vol. 9, no. 9, pp. 6650–6662, May 2022.

[152] Y. Chen, X. Wen, Y. Zhang, and Q. He, "FPC: Filter pruning via the contribution of output feature map for deep convolutional neural networks acceleration," *Knowl.-Based Syst.*, vol. 238, Feb. 2022, Art. no. 107876.

[153] S. Swaminathan, D. Garg, R. Kannan, and F. Andres, "Sparse low rank factorization for deep neural network compression," *Neurocomputing*, vol. 398, pp. 185–196, Jul. 2020.

[154] C. Shi, Z. Huang, L. Wan, and T. Xiong, "Low-rank tensor completion based on non-convex logDet function and Tucker decomposition," *Signal, Image Video Process.*, vol. 15, no. 6, pp. 1169–1177, Sep. 2021.

[155] Y. Fu, Y. Deng, Y. Zhang, Z. Yang, and R. Qi, "Low rank tucker decomposition for 2D+3D facial expression recognition," *Proc. Comput. Sci.*, vol. 198, pp. 499–504, Jan. 2022.

[156] L. Ma and E. Solomonik, "Fast and accurate randomized algorithms for low-rank tensor decompositions," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 24299–24312.

[157] A.-H. Phan, P. Tichavsky, K. Sobolev, K. Sozykin, D. Ermilov, and A. Cichocki, "Canonical polyadic tensor decomposition with low-rank factor matrices," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 4690–4694.

[158] P.-A. Chantal, A. Karfoul, A. Nica, and R. L. B. Jeannès, "Dynamic brain effective connectivity analysis based on low-rank canonical polyadic decomposition: Application to epilepsy," *Med. Biol. Eng. Comput.*, vol. 59, no. 5, pp. 1081–1098, May 2021.

[159] S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 2464–2469.

[160] T. Li, M. Xu, R. Tang, Y. Chen, and Q. Xing, "DeepQTMT: A deep learning approach for fast QTMT-based CU partition of intra-mode VVC," *IEEE Trans. Image Process.*, vol. 30, pp. 5377–5390, 2021.

[161] E. Park, D. Kim, S. Kim, Y.-D. Kim, G. Kim, S. Yoon, and S. Yoo, "Big/little deep neural network for ultra low power inference," in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Oct. 2015, pp. 124–132.

[162] V. S. Marco, B. Taylor, Z. Wang, and Y. Elkhatib, "Optimizing deep learning inference on embedded systems through adaptive model selection," *ACM Trans. Embedded Comput. Syst.*, vol. 19, no. 1, pp. 1–28, Jan. 2020.

[163] Z. Liu, C. Yang, J. Huang, S. Liu, Y. Zhuo, and X. Lu, "Deep learning framework based on integration of S-mask R-CNN and Inception-V3 for ultrasound image-aided diagnosis of prostate cancer," *Future Gener. Comput. Syst.*, vol. 114, pp. 358–367, Jan. 2021.

[164] D. Sarwinda, R. H. Paradisa, A. Bustamam, and P. Anggia, "Deep learning in image classification using residual network (ResNet) variants for detection of colorectal cancer," *Proc. Comput. Sci.*, vol. 179, pp. 423–431, Jan. 2021.

[165] K. D. Kadam, S. Ahirrao, K. Kotecha, and S. Sahu, "Detection and localization of multiple image splicing using MobileNet V1," *IEEE Access*, vol. 9, pp. 162499–162519, 2021.

[166] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan, "Cachier: Edge-caching for recognition applications," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 276–286.

[167] L. N. Huynh, Y. Lee, and R. K. Balan, "DeepMon: Mobile GPU-based deep learning framework for continuous vision applications," in *Proc. 15th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2017, pp. 82–95.

[168] A. Balasubramanian, A. Kumar, Y. Liu, H. Cao, S. Venkataraman, and A. Akella, "Accelerating deep learning inference via learned caches," 2021, *arXiv:2101.07344*.

[169] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Sci. China Technol. Sci.*, vol. 63, no. 10, pp. 1872–1897, 2020.

[170] A. Berthelier, T. Chateau, S. Duffner, C. Garcia, and C. Blanc, "Deep model compression and architecture optimization for embedded systems: A survey," *J. Signal Process. Syst.*, vol. 93, no. 8, pp. 863–878, Aug. 2021.

[171] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, Oct. 2021.

[172] S. Xu, A. Huang, L. Chen, and B. Zhang, "Convolutional neural network pruning: A survey," in *Proc. 39th Chin. Control Conf. (CCC)*, Jul. 2020, pp. 7458–7463.

[173] D. Gao, X. He, Z. Zhou, Y. Tong, K. Xu, and L. Thiele, "Rethinking pruning for accelerating deep inference at the edge," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 155–164.

[174] S. V. Naik, S. K. Majjigudda, S. Naik, S. M. Dandin, U. Kulkarni, S. M. Meena, S. V. Gurlahosur, and P. Benagi, "Survey on comparative study of pruning mechanism on MobileNetV3 model," in *Proc. Int. Conf. Intell. Technol. (CONIT)*, Jun. 2021, pp. 1–8.

[175] Y. Zhang, B. Li, and Y. Tan, "Making AI available for everyone at anywhere: A survey about edge intelligence," *J. Phys., Conf.*, vol. 1757, no. 1, Jan. 2021, Art. no. 012076.

[176] J. Wang, J. Zhang, W. Bao, X. Zhu, B. Cao, and P. S. Yu, "Not just privacy: Improving performance of private deep learning in mobile cloud," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2407–2416.

[177] R. Sharma, S. Biookaghazadeh, B. Li, and M. Zhao, "Are existing knowledge transfer techniques effective for deep learning with edge devices?" in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 42–49.

[178] S. Niu, M. Liu, Y. Liu, J. Wang, and H. Song, "Distant domain transfer learning for medical imaging," *IEEE J. Biomed. Health Informat.*, vol. 25, no. 10, pp. 3784–3793, Oct. 2021.

[179] D. Hazarika, S. Poria, R. Zimmermann, and R. Mihalcea, "Conversational transfer learning for emotion recognition," *Inf. Fusion*, vol. 65, pp. 1–12, Jan. 2021.

[180] E. Soleimani and E. Nazerfard, "Cross-subject transfer learning in human activity recognition systems using generative adversarial networks," *Neurocomputing*, vol. 426, pp. 26–34, Feb. 2021.

[181] E. Baccarelli, M. Scarpiniti, A. Momenzadeh, and S. S. Ahrabi, "Learning-in-the-fog (LiFo): Deep learning meets fog computing for the minimum-energy distributed early-exit of inference in delay-critical IoT realms," *IEEE Access*, vol. 9, pp. 25716–25757, 2021.

[182] X. Tan, H. Li, L. Wang, and Z. Xu, "End-edge coordinated inference for real-time BYOD malware detection using deep learning," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2020, pp. 1–6.

[183] N. Passalis, J. Raitoharju, A. Tefas, and M. Gabbouj, "Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits," *Pattern Recognit.*, vol. 105, Sep. 2020, Art. no. 107346.

[184] X. Tan, H. Li, L. Wang, X. Huang, and Z. Xu, "Empowering adaptive early-exit inference with latency awareness," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 11, pp. 9825–9833.

[185] S. Laskaridis, S. I. Venieris, H. Kim, and N. D. Lane, "HAPI: Hardware-aware progressive inference," in *Proc. 39th Int. Conf. Comput.-Aided Design*, Nov. 2020, pp. 1–9.

[186] A. V. Kumar and M. Sivathanu, "Quiver: An informed storage cache for deep learning," in *Proc. 18th USENIX Conf. File Storage Technol.*, 2020, pp. 283–296.

[187] N. Krichevsky, R. S. Louis, and T. Guo, "Quantifying and improving performance of distributed deep learning with cloud storage," in *Proc. IEEE Int. Conf. Cloud Eng. (ICE)*, Oct. 2021, pp. 99–109.

[188] E. Romero-Gainza, C. Stewart, A. Li, K. Hale, and N. Morris, "Memory mapping and parallelizing random forests for speed and cache efficiency," in *Proc. 50th Int. Conf. Parallel Process. Workshop*, Aug. 2021, pp. 1–5.

[189] S.-T. Cheng, C.-W. Hsu, G.-J. Horng, and C.-H. Lin, "Adaptive cache pre-forwarding policy for distributed deep learning," *Comput. Electr. Eng.*, vol. 82, Mar. 2020, Art. no. 106558.

[190] G. Zong, Q. Li, P. Zhang, G. Zhang, and X. Zhu, "Efficient cache strategy for face recognition system," in *Proc. 5th Int. Conf. Big Data Comput.*, May 2020, pp. 108–113.

[191] L. Wang, Q. Luo, and S. Yan, "DIESEL+: Accelerating distributed deep learning tasks on image datasets," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 5, pp. 1173–1184, May 2022.

[192] A. F. Inci, M. M. Isgenc, and D. Marculescu, "DeepNVM: A framework for modeling and analysis of non-volatile memory technologies for deep learning applications," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 1295–1298.

[193] T. A. Khoa, D.-V. Nguyen, M.-S. Dao, and K. Zettsu, "Fed xData: A federated learning framework for enabling contextual health monitoring in a cloud-edge network," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2021, pp. 4979–4988.

[194] G. Yang, B. Wang, S. Qiao, L. Qu, N. Han, G. Yuan, H. Li, T. Wu, and Y. Peng, "Distilled and filtered deep neural networks for real-time object detection in edge computing," *Neurocomputing*, vol. 505, pp. 225–237, Sep. 2022.

[195] E. Kristiani, C.-T. Yang, and C.-Y. Huang, "ISEC: An optimized deep learning model for image classification on edge computing," *IEEE Access*, vol. 8, pp. 27267–27276, 2020.

[196] V. K. Singh and M. H. Kolekar, "Deep learning empowered COVID-19 diagnosis using chest CT scan images for collaborative edge-cloud computing platform," *Multimedia Tools Appl.*, vol. 81, no. 1, pp. 3–30, 2021.

[197] K. Yu, L. Tan, L. Lin, X. Cheng, Z. Yi, and T. Sato, "Deep-learning-empowered breast cancer auxiliary diagnosis for 5GB remote E-health," *IEEE Wireless Commun.*, vol. 28, no. 3, pp. 54–61, Jul. 2021.

[198] M. Kumar, V. Shenbagaraman, R. N. Shaw, and A. Ghosh, "Predictive data analysis for energy management of a smart factory leading to sustainability," in *Innovations in Electrical and Electronic Engineering*. Berlin, Germany: Springer, 2021, pp. 765–773.

[199] J. Violos, T. Pagoulatou, S. Tsanakas, K. Tserpes, and T. Varvarigou, "Predicting resource usage in edge computing infrastructures with CNN and a hybrid Bayesian particle swarm hyper-parameter optimization model," in *Intelligent Computing*. Berlin, Germany: Springer, 2021, pp. 562–580.

[200] A. S. Shitole and M. H. Devare, "Optimization of IoT-enabled physical location monitoring using DT and VAR," *Int. J. Cognit. Informat. Natural Intell.*, vol. 15, no. 4, pp. 1–28, Oct. 2021.

[201] A. S. Moursi, N. El-Fishawy, S. Djahel, and M. A. Shouman, "An IoT enabled system for enhanced air quality monitoring and prediction on the edge," *Complex Intell. Syst.*, vol. 7, no. 6, pp. 2923–2947, 2021.

[202] U. Farooq, M. W. Shabir, M. A. Javed, and M. Imran, "Intelligent energy prediction techniques for fog computing networks," *Appl. Soft Comput.*, vol. 111, Nov. 2021, Art. no. 107682.

[203] X. Wang, T. Wei, L. Kong, L. He, F. Wu, and G. Chen, "ECASS: Edge computing based auxiliary sensing system for self-driving vehicles," *J. Syst. Archit.*, vol. 97, pp. 258–268, Aug. 2019.

[204] Z. Gao, H. Zhang, S. Dong, S. Sun, X. Wang, G. Yang, W. Wu, S. Li, and V. H. C. De Albuquerque, "Salient object detection in the distributed cloud-edge intelligent network," *IEEE Netw.*, vol. 34, no. 2, pp. 216–224, Mar. 2020.

[205] E. Figetakis and A. Refaey, "UAV path planning using on-board ultrasound transducer arrays and edge support," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2021, pp. 1–6.

[206] P. Zhu, J. Xu, J. Li, D. Wang, and X. You, "Learning-empowered privacy preservation in beyond 5G edge intelligence networks," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 12–18, Apr. 2021.

[207] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6G wireless system: Design challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 40–47, Apr. 2021.

[208] J. Shao, Y. Mao, and J. Zhang, "Task-oriented communication for multidevice cooperative edge inference," *IEEE Trans. Wireless Commun.*, early access, Jul. 22, 2022, doi: 10.1109/TWC.2022.3191118.

[209] Y. Liu, Y. Zhu, and J. James, "Resource-constrained federated edge learning with heterogeneous data: Formulation and analysis," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3166–3178, Sep. 2022.

[210] X. Li, S. Chen, Y. Zhou, J. Chen, and G. Feng, "Intelligent service migration based on hidden state inference for mobile edge computing," *IEEE Trans. Cognit. Commun. Netw.*, vol. 8, no. 1, pp. 380–393, Mar. 2022.

[211] A. Du, Y. Shen, Q. Zhang, L. Tseng, and M. Aloqaily, "CRACAU: Byzantine machine learning meets industrial edge computing in Industry 5.0," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5435–5445, Aug. 2022.

[212] O. Fagbohungbe, S. R. Reza, X. Dong, and L. Qian, "Efficient privacy preserving edge intelligent computing framework for image classification in IoT," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 4, pp. 941–956, Aug. 2022.

[213] Y. Deng, F. Lyu, J. Ren, Y. Zhang, Y. Zhou, Y. Zhang, and Y. Yang, "SHARE: Shaping data distribution at edge for communication-efficient hierarchical federated learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 24–34.

[214] O. Goldstein, M. Kachuee, and M. Sarrafzadeh, "Decentralized knowledge transfer on edge networks for detecting cancer in images," in *Proc. IEEE EMBS Int. Conf. Biomed. Health Informat. (BHI)*, Jul. 2021, pp. 1–5.

[215] C. Sun, X. Wu, X. Li, Q. Fan, J. Wen, and V. C. M. Leung, "Cooperative computation offloading for multi-access edge computing in 6G mobile networks via soft actor critic," *IEEE Trans. Netw. Sci. Eng.*, early access, Apr. 30, 2021, doi: 10.1109/TNSE.2021.3076795.

[216] Z. Guo, M. Liu, Z. Yuan, L. Shen, W. Liu, and T. Yang, "Communication-efficient distributed stochastic AUC maximization with deep neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3864–3874.

[217] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[218] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8099–8110, Sep. 2020.

[219] A. Nedic, "Distributed gradient methods for convex machine learning problems in networks: Distributed optimization," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 92–101, May 2020.

[220] J. Jiang, F. Fu, T. Yang, Y. Shao, and B. Cui, "SKCompress: Compressing sparse and nonuniform gradient in distributed machine learning," *VLDB J.*, vol. 29, no. 5, pp. 945–972, Sep. 2020.

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

[222] B. Yang, O. Fagbohungbe, X. Cao, C. Yuen, L. Qian, D. Niyato, and Y. Zhang, "A joint energy and latency framework for transfer learning over 5G industrial edge networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 531–541, Jan. 2022.

[223] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 7947–7962, Dec. 2021.

[224] Y. Liu, L. Kong, G. Chen, F. Xu, and Z. Wang, "Light-weight AI and IoT collaboration for surveillance video pre-processing," *J. Syst. Archit.*, vol. 114, Mar. 2021, Art. no. 101934.

[225] M. Li, J. Gao, C. Zhou, X. S. Shen, and W. Zhuang, "Slicing-based artificial intelligence service provisioning on the network edge: Balancing AI service performance and resource consumption of data management," *IEEE Veh. Technol. Mag.*, vol. 16, no. 4, pp. 16–26, Dec. 2021.

[226] Y. Zhu, "Network public opinion prediction and control based on edge computing and artificial intelligence new paradigm," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–11, Apr. 2021.

[227] N. Shlezinger, E. Farhan, H. Morgenstern, and Y. C. Eldar, "Collaborative inference via ensembles on the edge," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 8478–8482.

[228] Y. Chen, C. Hawkins, K. Zhang, Z. Zhang, and C. Hao, "3U-EdgeAI: Ultra-low memory training, ultra-low bitwidth quantization, and ultra-low latency acceleration," in *Proc. Great Lakes Symp. VLSI*, Jun. 2021, pp. 157–162.

[229] W. Y. B. Lim, J. S. Ng, Z. Xiong, J. Jin, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 536–550, Mar. 2022.

[230] L. Welagedara, J. Harischandra, and N. Jayawardene, "A review on edge intelligence based collaborative learning approaches," in *Proc. IEEE 11th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2021, pp. 0572–0577.

[231] P. Janakaraj, "Towards AI-empowered wireless networks: From edge to core," Univ. North Carolina Charlotte, Charlotte, NC, USA, Tech. Rep. 28490960, 2021.

[232] N. Tonellotto, A. Gotta, F. M. Nardini, D. Gadler, and F. Silvestri, "Neural network quantization in federated learning at the edge," *Inf. Sci.*, vol. 575, pp. 417–436, Oct. 2021.

[233] I. Wagner and D. Eckhoff, "Technical privacy metrics: A systematic survey," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 1–38, Jun. 2018.

[234] K. A. Vaibhavi, "Survey on cryptographic algorithms in cloud computing," *J. Res. Proc.*, vol. 1, no. 2, pp. 53–77, 2021.

[235] N. Subbanna, M. Wilms, A. Tuladhar, and N. D. Forkert, "An analysis of the vulnerability of two common deep learning-based medical image segmentation techniques to model inversion attacks," *Sensors*, vol. 21, no. 11, p. 3874, Jun. 2021.

[236] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer, "Membership inference attacks from first principles," in *Proc. IEEE Symp. Secur. Privacy*, May 2022, pp. 1897–1914.

[237] R. D. Georgiev, "Analysis of deep learning inference compute and energy consumption trends," Escuela Técnica Superior de Ingeniería Informática ETSINF, Universitat Politècnica de València, Valencia, Spain, 2021.

[238] N. Nez, A. N. Vilchez, H. R. Zohouri, O. Khavin, and S. Dasgupta, "Dynamic neural accelerator for reconfigurable & energy-efficient neural network inference," in *Proc. IEEE Hot Chips Symp. (HCS)*, Aug. 2021, pp. 1–21.

[239] C.-S. Mei and C. Wang, "Energy efficiency and timeliness in model training for Internet-of-Things applications," in *Proc. Int. Conf. Internet-Things Design Implement.*, May 2021, pp. 253–254.

[240] S. Zhu, K. Ota, and M. Dong, "Green AI for IIoT: Energy efficient intelligent edge computing for industrial Internet of Things," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 79–88, Mar. 2022.

[241] Q. Liang, P. Shenoy, and D. Irwin, "AI on the edge: Characterizing AI-based IoT applications using specialized edge architectures," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Oct. 2020, pp. 145–156.

[242] G. Flamis, S. Kalapothas, and P. Kitsos, "Best practices for the deployment of edge inference: The conclusions to start designing," *Electronics*, vol. 10, no. 16, p. 1912, Aug. 2021.

[243] B. Varghese, N. Wang, D. Bermbach, C.-H. Hong, E. D. Lara, W. Shi, and C. Stewart, "A survey on edge performance benchmarking," *ACM Comput. Surveys*, vol. 54, no. 3, pp. 1–33, Apr. 2022.

[244] T.-C. Chen, W.-T. Wang, K. Kao, C.-L. Yu, C. Lin, S.-H. Chang, and P.-K. Tsung, "NeuroPilot: A cross-platform framework for edge-AI," in *Proc. IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Mar. 2019, pp. 167–170.

[245] M. Merenda, C. Porcaro, and D. Iero, "Edge machine learning for AI-enabled IoT devices: A review," *Sensors*, vol. 20, no. 9, p. 2533, 2020.

[246] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, "Bringing AI to edge: From deep learning's perspective," *Neurocomputing*, vol. 485, pp. 297–320, May 2022.

[247] E. Cai, D.-C. Juan, D. Stamoulis, and D. Marculescu, "NeuralPower: Predict and deploy energy-efficient convolutional neural networks," in *Proc. 9th Asian Conf. Mach. Learn.*, vol. 77, M.-L. Zhang and Y.-K. Noh, Eds. Seoul, Republic of Korea: Yonsei University, Nov. 2017, pp. 622–637. [Online]. Available: https://proceedings.mlr.press/v77/cai17a.html

[248] F. Yu, Z. Qin, C. Liu, L. Zhao, Y. Wang, and X. Chen, "Interpreting and evaluating neural network robustness," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4199–4205.

[249] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 656–672.

[250] J. Jiang, Y. Shu, J. Wang, and M. Long, "Transferability in deep learning: A survey," 2022, *arXiv:2201.05867*.

[251] C. V. Nguyen, T. Hassner, M. Seeger, and C. Archambeau, "Leep: A new measure to evaluate transferability of learned representations," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 7294–7305.

[252] P. Bhatt, S. Sarangi, and S. Pappula, "Comparison of CNN models for application in crop health assessment with participatory sensing," in *Proc. IEEE Global Humanitarian Technol. Conf. (GHTC)*, Oct. 2017, pp. 1–7.

[253] S. Wang, D. Li, Y. Cheng, J. Geng, Y. Wang, S. Wang, S.-T. Xia, and J. Wu, "BML: A high-performance, low-cost gradient synchronization algorithm for DML training," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.

[254] X. Li, D. Yang, Y. Wang, S. Yang, L. Qi, F. Li, Z. Gan, and W. Zhang, "Automatic tongue image segmentation for real-time remote diagnosis," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Nov. 2019, pp. 409–414.

[255] M. Khani, P. Hamadanian, A. Nasr-Esfahany, and M. Alizadeh, "Real-time video inference on edge devices via adaptive model streaming," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 4572–4582.

[256] H. E. Ilhan, S. Ozer, G. K. Kurt, and H. Ali Cirpan, "Offloading deep learning empowered image segmentation from UAV to edge server," in *Proc. 44th Int. Conf. Telecommun. Signal Process. (TSP)*, Jul. 2021, pp. 296–300.

[257] J. K. Mahendran, D. T. Barry, A. K. Nivedha, and S. M. Bhandarkar, "Computer vision-based assistance system for the visually impaired using mobile edge artificial intelligence," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 2418–2427.

[258] M. A. Rahman and M. S. Hossain, "An internet-of-medical-things-enabled edge computing framework for tackling COVID-19," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15847–15854, Nov. 2021.

[259] A. George, A. Ravindran, M. Mendieta, and H. Tabkhi, "MEZ: An adaptive messaging system for latency-sensitive multi-camera machine vision at the IoT edge," *IEEE Access*, vol. 9, pp. 21457–21473, 2021.

[260] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, "A state-of-the-art survey on solving non-IID data in federated learning," *Future Gener. Comput. Syst.*, vol. 135, pp. 244–258, Oct. 2022.

[261] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[262] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, and R. G. D'Oliveira, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[263] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consum. Electron. Mag.*, vol. 9, no. 3, pp. 8–16, May 2020.

[264] M. A. Rahman and M. S. Hossain, "A deep learning assisted software defined security architecture for 6G wireless networks: IIoT perspective," *IEEE Wireless Commun.*, vol. 29, no. 2, pp. 52–59, Apr. 2022.

[265] R. Shokri, M. Strobel, and Y. Zick, "On the privacy risks of model explanations," in *Proc. AAAI/ACM Conf. AI, Ethics, Soc.*, Jul. 2021, pp. 231–241.

[266] H.-Y. Tran and J. Hu, "Privacy-preserving big data analytics a comprehensive survey," *J. Parallel Distrib. Comput.*, vol. 134, pp. 207–218, Dec. 2019.

[267] G. Kaissis, A. Ziller, J. Passerat-Palmbach, T. Ryffel, D. Usynin, A. Trask, I. Lima, J. Mancuso, F. Jungmann, M. M. Steinborn, and A. Saleh, "End-to-end privacy preserving deep learning on multi-institutional medical imaging," *Nature Mach. Intell.*, vol. 3, pp. 473–484, Jun. 2021.

[268] M. Alkhelaiwi, W. Boulila, J. Ahmad, A. Koubaa, and M. Driss, "An efficient approach based on privacy-preserving deep learning for satellite image classification," *Remote Sens.*, vol. 13, no. 11, p. 2221, Jun. 2021.

[269] X. Zhang, J. Ding, M. Wu, S. T. C. Wong, H. Van Nguyen, and M. Pan, "Adaptive privacy preserving deep learning algorithms for medical data," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 1169–1178.

[270] N. Jain, K. Nandakumar, N. Ratha, S. Pankanti, and U. Kumar, "PPDL—Privacy preserving deep learning using homomorphic encryption," in *Proc. 5th Joint Int. Conf. Data Sci. Manag. Data*, Jan. 2022, pp. 318–319.

[271] S. Tan, B. Knott, Y. Tian, and D. J. Wu, "CryptGPU: Fast privacy-preserving machine learning on the GPU," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 1021–1038.

[272] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1608–1631, Aug. 2019.

[273] W. Xiong, E. Legrand, O. Åberg, and R. Lagerström, "Cyber security threat modeling based on the MITRE enterprise ATT&CK matrix," *Softw. Syst. Model.*, vol. 21, no. 1, pp. 157–177, Feb. 2022.

[274] D. Pasquini, G. Ateniese, and M. Bernaschi, "Unleashing the tiger: Inference attacks on split learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 2113–2129, doi: 10.1145/3460120.3485259.

[275] F. Al-Doghman, N. Moustafa, I. Khalil, Z. Tari, and A. Zomaya, "AI-enabled secure microservices in edge computing: Opportunities and challenges," *IEEE Trans. Services Comput.*, early access, Mar. 1, 2022, doi: 10.1109/TSC.2022.3155447.

[276] W. Yang, W. Liu, X. Wei, Z. Guo, K. Yang, H. Huang, and L. Qi, "EdgeKeeper: A trusted edge computing framework for ubiquitous power Internet of Things," *Frontiers Inf. Technol. Electron. Eng.*, vol. 22, no. 3, pp. 374–399, Mar. 2021.

[277] P.-Y. Gong, C.-H. Wang, J.-P. Sheu, and D.-N. Yang, "Distributed DRL-based resource allocation for multicast D2D communications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 01–06.

[278] M. Ezzeddine, R. Morcel, R. Artail, M. A. R. Saghir, H. Akkary, and H. Hajj, "RESTful hardware microservices using reconfigurable networked accelerators in cloud and edge datacenters," in *Proc. IEEE 7th Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2018, pp. 1–4.

[279] Y. Xiao, G. Shi, Y. Li, W. Saad, and H. V. Poor, "Toward self-learning edge intelligence in 6G," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 34–40, Dec. 2020.

[280] Y. An, J. Li, F. R. Yu, J. Chen, and V. C. M. Leung, "A novel HTTP anomaly detection framework based on edge intelligence for the Internet of Things (IoT)," *IEEE Wireless Commun.*, vol. 28, no. 2, pp. 159–165, Apr. 2021.

[281] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. Workshop Mobile Edge Commun.*, Aug. 2018, pp. 31–36.

[282] L. Liu, J. Chen, M. Brocanelli, and W. Shi, "E2M: An energy-efficient middleware for computer vision applications on autonomous mobile robots," in *Proc. 4th ACM/IEEE Symp. Edge Comput.*, Nov. 2019, pp. 59–73.

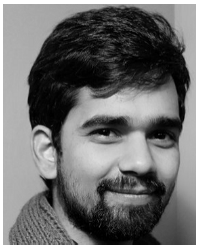
[283] X. Zhang, Y. Wang, S. Lu, L. Liu, L. Xu, and W. Shi, "OpenEI: An open framework for edge intelligence," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 1840–1851.

[284] H. Hashemi, Y. Wang, and M. Annavaram, "DarKnight: An accelerated framework for privacy and integrity preserving deep learning using trusted hardware," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2021, pp. 212–224.

[285] F. Shehzad, M. Rashid, M. H. Sinky, S. S. Alotaibi, and M. Y. I. Zia, "A scalable system-on-chip acceleration for deep neural networks," *IEEE Access*, vol. 9, pp. 95412–95426, 2021.

[286] K. S. Zaman, M. B. I. Reaz, S. H. M. Ali, A. A. A. Bakar, and M. E. H. Chowdhury, "Custom hardware architectures for deep learning on portable devices: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6068–6088, Nov. 2022.

[287] S. Mittal and S. Umesh, "A survey on hardware accelerators and optimization techniques for RNNs," *J. Syst. Archit.*, vol. 112, Jan. 2021, Art. no. 101839.

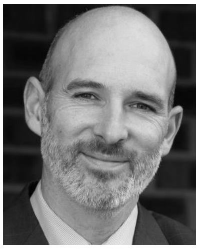


PRAVEEN JOSHI is currently pursuing the Ph.D. degree under the supervision of Dr. Haithem Afli, Dr. Mohammed Hasanuzzaman, and Dr. Ted Scully from Computer Science Department, Munster Technological University through the support of the Advanced CRT. His master's thesis in artificial intelligence studies dealt with stock market forecasting through a multi-modal machine learning approach by exploring the semantics of social platforms and business news. The financial modal build achieved nice accuracy and at the same time overcome the explainability aspect of the deep learning model by comprehending the visualization techniques to represent the events happening in daily-day life. In continuation with such research interests, and a commitment to critical and self-reflexive modes of knowledge production, his Ph.D. endeavour aims to be a researcher of a sustainable society in the coming future.

MOHAMMED HASANUZZAMAN is currently a Lecturer with the Department of Computer Science, Munster Technological University, Ireland. He is also a member of the ADAPT Centre-Ireland's global center of excellence for digital content and media innovation and funded investigator of the Science Foundation Ireland, Ireland.

CHANDRA THAPA (Member, IEEE) received the Ph.D. degree from The University of Newcastle, Australia, in 2018. He is currently a Research Scientist at CSIRO Data61. His research interests include privacy-preserving computation, distributed systems security, and network information theory. His current works include data security and privacy, application of privacy-preserving approaches to machine learning/artificial intelligence, and cybersecurity for AI.

HAITHEM AFLI is a Leading Expert in natural language processing and applied artificial intelligence in healthcare, life-science, and fintech. He is lecturing AI with the Department of Computer Science, Munster Technological University (MTU), Ireland, and leading the MTU Human Centered AI Research Group. He is a Funded Investigator with Science Foundation Ireland with the ADAPT Centre, where he is a member of the ADAPT Executive Management Committee, representing MTU, and also an Academic Researcher, he is keen to commercialize his research with industry partnerships and is actively involved in managing academia-industry partnership projects. His research interests include machine translation, sentiment analysis, natural language processing, and machine learning. He is serving as an Editor, the Program Chair, a Program Committee Member and an Advisor for many international research conferences and journals.



TED SCULLY received the B.Sc. degree in computer science from University College Cork and the Ph.D. degree in artificial intelligence (AI) from the National University of Ireland, Galway. He is currently a Senior Lecturer with the Department of Computer Science, Munster Technological University. He is also a Principal Investigator at the Ríomh Research Group and the coordinator of the M.Sc. students in AI. He has both led and participated in a range of multifaceted research and development projects focusing on the application of optimization techniques and machine learning algorithms to a diverse set of problem domains, such as supply chain optimization, load balancing in wireless networks, micro-grid management, and water and energy modeling on Irish dairy farms. His research interest include machine learning/DL and meta-heuristic optimization.