KIPS Trans. Comp. and Comm. Sys.
Vol.9, No.12 pp.291~306
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2020.9.12.291

Distributed Edge Computing for DNA-Based Intelligent Services and Applications: A Review

Temesgen Seyoum Alemayehu† · We-Duke Cho††

ABSTRACT

Nowadays, Data-Network-AI (DNA)-based intelligent services and applications have become a reality to provide a new dimension of services that improve the quality of life and productivity of businesses. Artificial intelligence (AI) can enhance the value of IoT data (data collected by IoT devices). The internet of things (IoT) promotes the learning and intelligence capability of AI. To extract insights from massive volume IoT data in real-time using deep learning, processing capability needs to happen in the IoT end devices where data is generated. However, deep learning requires a significant number of computational resources that may not be available at the IoT end devices. Such problems have been addressed by transporting bulks of data from the IoT end devices to the cloud datacenters for processing. But transferring IoT big data to the cloud incurs prohibitively high transmission delay and privacy issues which are a major concern. Edge computing, where distributed computing nodes are placed close to the IoT end devices, is a viable solution to meet the high computation and low-latency requirements and to preserve the privacy of users. This paper provides a comprehensive review of the current state of leveraging deep learning within edge computing to unleash the potential of IoT big data generated from IoT end devices. We believe that the revision will have a contribution to the development of DNA-based intelligent services and applications. It describes the different distributed training and inference architectures of deep learning models across multiple nodes of the edge computing platform. It also provides the different privacy-preserving approaches of deep learning on the edge computing environment and the various application domains where deep learning on the network edge can be useful. Finally, it discusses open issues and challenges leveraging deep learning within edge computing.

Keywords : IoT, Deep Learning, Edge Computing, Distributed Training, DNA

딥러닝을 사용하는 IoT빅데이터 인프라에 필요한 DNA 기술을 위한 분산 엣지 컴퓨팅기술 리뷰

Temesgen Seyoum Alemayehu† · 조 위 덕††

요 약

오늘날 데이터 네트워크 AI (DNA) 기반 지능형 서비스 및 애플리케이션은 비즈니스의 삶의 질과 생산성을 향상시키는 새로운 차원의 서비스를 제공하는 것이 현실이 되었다. 인공지능(AI)은 IoT 데이터(IoT 장치에서 수집한 데이터)의 가치를 높이며, 사물 인터넷(IoT)은 AI의 학습 및 지능 기능을 촉진한다. 딥러닝을 사용하여 대량의 IoT 데이터에서 실시간으로 인사이트를 추출하려면 데이터가 생성되는 IoT 단말 장치에서의 처리 능력이 필요하다. 그러나 딥러닝에는 IoT 최종 장치에서 사용할 수 없는 상당 수의 컴퓨팅 리소스가 필요하다. 이러한 문제는 처리를 위해 IoT 최종 장치에서 클라우드 데이터 센터로 대량의 데이터를 전송함으로써 해결되었다. 그러나 IoT 빅 데이터를 클라우드로 전송하면 엄청나게 높은 전송 지연과 주요 관심사인 개인 정보 보호 문제가 발생한다. 분산 컴퓨팅 노드가 IoT 최종 장치 가까이에 배치되는 엣지 컴퓨팅은 높은 계산 및 짧은 지연 시간 요구 사항을 충족하고 사용자의 개인 정보를 보호하는 실행 가능한 솔루션이다. 본 논문에서는 엣지 컴퓨팅 내에서 딥러닝을 활용하여 IoT 최종 장치에서 생성된 IoT 빅 데이터의 잠재력을 발휘하는 현재 상태에 대한 포괄적인 검토를 제공한다. 우리는 이것이 DNA 기반 지능형 서비스 및 애플리케이션 개발에 기여할 것이라고 본다. 엣지 컴퓨팅 플랫폼의 여러 노드에서 딥러닝 모델의 다양한 분산 교육 및 추론 아키텍처를 설명하고 엣지 컴퓨팅 환경과 네트워크 엣지에서 딥러닝이 유용할 수 있는 다양한 애플리케이션 도메인에서 딥러닝의 다양한 개인 정보 보호 접근 방식을 제공한다. 마지막으로 엣지 컴퓨팅 내에서 딥러닝을 활용하는 열린 문제와 과제에 대해 설명한다.

키워드 : IoT, 딥러닝, 엣지컴퓨팅, 분산훈련, DNA

※ 이 논문은 한국연구재단 이공분야기초연구 과제의 지원으로 수행되었음 (2019063128).
† 비 회 원 : 아주대학교 라이프케어사이언스랩 연구교수
†† 종신회원 : 아주대학교 전자공학부 교수

Manuscript Received : September 18, 2020
First Revision : November 16, 2020
Accepted : December 9, 2020
* Corresponding Author : We-Duke Cho(chowd@ajou.ac.kr)

1. Introduction

The amount of data generated by Internet of things (IoT) devices grows as the number of connected IoT devices increases at steady pace. In the year 2025, a new forecast from International Data Corporation (IDC) [1] approximates that there will be 41.6 billion connected IoT devices that generate 79.4 zettabytes (ZB) of data. The IoT big data, generated daily by IoT end devices, put great demands on data processing to solve large-scale complex problems. Analyzing this massive IoT data intelligently will play a significant role for DNA-based intelligent services and applications. IoT, IoT big data and artificial intelligence (AI) are the three digital pillars of DNA-based intelligent services and applications. They improve the quality of life and productivity of industries as AI based analysis becomes critical to extract insights from IoT big data in real time [2]. Nowadays, deep learning as special case of AI is widely deployed to a variety of business sectors to improve the quality of life and productivity of businesses. For instance, applications, such as automotive [3-5], smart cities [6-8], health care [9-11], computer vision and natural language processing [12, 13], etc., are primarily driven by the IoT big data, machine language algorithms, high-performance computation and storage facilities.

While DNA has brought unforeseen opportunities for various applications, there are also several architectural and data deluge challenges while deploying DNA based intelligent applications in practice. On-device analyses of IoT big data on the device itself (locally) suffer from poor performance due to resource constraints. Extracting insights from the IoT big data requires a significant amount of computation resources and storage facilities which may not be available at the IoT end devices. Such challenges have been addressed by transporting the data bulks from the IoT devices to the cloud datacenters for processing [14]. However, transferring IoT big data to the cloud incurs prohibitively high transmission delay and privacy problems which are a major concern [15]. The delay to access cloud services might not be short enough to satisfy the requirements of time-critical applications like cooperative autonomous driving [5]. Applications, such as camera frames of an autonomous vehicle require real-time inferences to detect and avoid obstacles. Sending private data to the cloud risks privacy issues which are critical to areas such as smart homes and cities. In addition to that, sending data from the IoT devices to the cloud introduces scalability problems. As the number of connected IoT devices increases, the volume of data increases, and network access to the cloud can become a bottleneck due to bandwidth constraints.

The various challenging issues, such as the high transmission delay, privacy, and scalability problems in cloud computing have led to a new distributed computing architecture called edge computing [16,17]. In Edge computing, some parts of the computing tasks can be done cooperatively at the distributed edge devices or IoT devices rather than having everything computed in the cloud (see Fig. 1). Detail research works on edge computing can be found in [18, 19]. Since the edge is closer to the IoT devices than the cloud, edge computing is a viable solution to resolve the latency, privacy, and scalability challenges to have high-performance AI for analyzing huge IoT data. Edge computing performs substantial computing close to the IoT devices to minimize data transmission and response time [20]. The edge computing's proximity to IoT end devices decreases the end-to-end latency and enables to provide real-time DNA-based intelligent services and applications. Edge computing addresses scalability issues as it enables a hierarchical architecture of IoT end devices, edge computing devices, and cloud data centers avoiding network bottlenecks at a central location. Edge computing also preserves the privacy of users as it avoids traversal of the public Internet and enables data to be analyzed by local trusted edge nodes close to the IoT end devices.

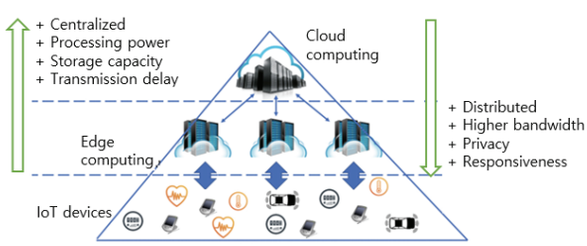

Fig. 1. Edge Computing vs Cloud Computing

The rest of the paper is organized as follows: Section 2 presents deep learning and the three most widely applied DNN structures. Section 3 addresses the two main enhancements that enable deep learning to run in the edge computing platform. Section 4 presents the two main types of distributed DNN training: Data parallelism and Model parallelism. Section 5 provides the different privacy preserving approaches of deep learning on the edge computing environment. Section 6 presents the various application domains where deep learning on the network edge can be useful. Section 7 discusses open issues and challenges leveraging deep learning within edge computing and finally, Section 8 concludes the paper.

2. Deep Learning

Deep learning is a subfield of the family of machine learning based on the structure and function of the brain called artificial neural networks (ANN) with representation learning. More details about deep learning can be found in [21, 22]. It is the most promising approach to resolve the problem of reliably extracting real-world IoT data from a complex environment that confuses the conventional machine learning techniques [23]. ANN with multiple layers between the input and output layers is called deep neural network (DNN) [22]. The three most widely applied DNN structures are Multilayer Perceptrons (MLPs), Convolution Neural Network (CNN), and Recurrent Neural Network (RNN) (See Fig. 2). Multilayer perceptron (MLPs) is a fully connected neural network of simple neurons called perceptron stacked in several layers to solve complex computational problems from a massive volume of the dataset [24]. Each perceptron in the input layer sends outputs to all the perceptrons in the hidden layers and all perceptrons in the hidden layer send outputs to the output layer. MLPs are used for problems related to classification prediction where inputs are assigned a label. MLPs are also used for problems of regression prediction where a real-valued quantity is predicted from inputs. The convolutional neural networks (CNNs) are special cases of DNNs that involve the usage of the matrix multiplications with convolutional filter operations. CNNs are suitable for data that has a spatial relationship which is important when working with images. They are common in DNNs which are designed for image and video analysis [25,26]. Recurrent neural networks (RNNs) are designed especially for time series prediction [27, 26], which are characterized by having loops in their layer connections to keep state and enable predictions on sequential inputs. They are suitable to work with prediction problems related to sequences such as sequences of words in a sentence for natural language processing or sequence of sounds in speech processing or recognition.

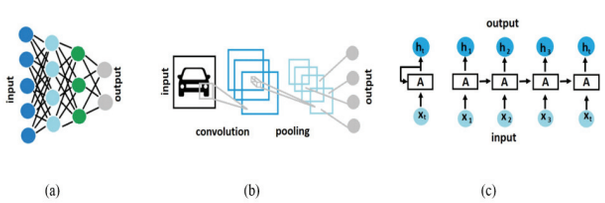

Fig. 2. The Three Most Widely Applied DNN Structures [26]: (a) Multilayer Perceptron, (b) Convolutional Neural Networks, (c) Recurrent Neural Networks

3. Deep Learning in the Edge Computing

IoT end devices are generating data that need to be analyzed in real time using deep learning. However, the training and inference of DNN require high computation resources to run quickly which may not be available at the IoT end devices. Edge computing is relatively smaller storage capability and limited power supply in edge devices as compared to the cloud datacenters. Due to this fact, deep learning in the edge computing has been enabled by two main improvements: The first one is to design an efficient deep learning model that reduce computational and memory space requirements to train the model with less time. The other one is to distribute the training and inference tasks of deep learning among the IoT end devices, the edge data servers and the cloud datacenters for parallel processing to obtain better efficiency.

3.1 Efficient Deep Learning Model for Reducing Computational and Memory Requirements

The full burden of training a DNN model is too intensive for a single resource constrained IoT end devices as it requires abundant memory and high computational power to store the training data and to train the model, respectively. Deploying DNN on IoT end devices such as mobile phones, vehicles, medical devices, drones or surveillance cameras, etc., remains a big challenge. To address the challenges, various lightweight models that require low processing power which can be deployed on IoT end devices have been designed to operate efficiently on edge devices [28]. Model compression is one of the techniques of creating lightweight models that can allow resource hungry DNN model to run on tiny IoT end devices [29]. It reduces the processing and memory requirements of DNNs with minimal effect on the overall accuracy of the model. In the literature, various DNN model compression techniques including data quantization, pruning, knowledge distillation, etc. have been proposed.

1) Quantization

Quantization is one of the model compression technique that satisfy the extreme memory and the higher processing power demands of DNN models at the expense of a minimal loss in accuracy [30, 31]. It reduces the complexity of the number of bits (bitwidth) and arithmetic operators to represent parameters and activations. Bitwidth that represent data determines the complexity of opera-

tors. Operators for floating-point arithmetic are more complex compared to integer arithmetic or fixed- point. The default 32-bit floating type variables that represent model weights have two main problems: The 32-bit floating point format that represent each weight requires considerable memory and makes the model very large. In addition to that the execution of the 32 float type variables is slow compared to the more compact 16-bit or 8-bit type variables that represent weights. To address such problems, quantization converts DNN parameters from 32-bit floating point towards 16-bit or 8-bit models, even to 1-bit. It effectively compacts the model size and accelerate the training and inference actions. A model that use 8-bit fixed-point data representations in [32] achieved an accuracy close to that obtained by the same model using 32 bit floating-points.

2) Pruning

DNN Pruning is an elimination of connections (synapses) and or neurons that are not useful or redundant to the DNN to reduce computational and memory demands (See Fig. 3) [33]. Several training techniques that apply pruning on the pre-trained network have been proposed in [34, 35]. Pruning improves performance and energy efficiency as it provides a smaller and faster network [36, 37].

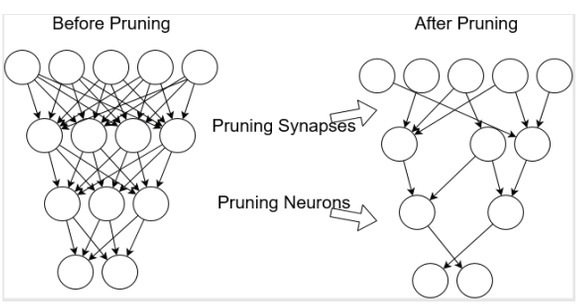

Fig. 3. Synapsis and Neurons Before and After Pruning [33]

The study in [38] presents a pruning method that compresses commonly used deep learning structures, such as convolutional, and recurrent neural networks in IoT end devices for sensing applications. The pruned DNN model can directly use the existing deep learning libraries to be deployed on edge devices without further modification. An effective energy-aware pruning method that uses the energy consumption of the deep convolutional neural network to guide the pruning process has been proposed in [39]. The proposed pruning method reduces the energy consumption of AlexNet and GoogLeNet, by 3.7× and 1.6×, respectively, with less than 1% top-5 accuracy loss.

3) Knowledge Distillation

Knowledge distillation is one of model compression technique that provides smaller models (named as student) that solve the same task as the larger models (named as teacher) [40]. This technique involves creating a smaller DNN model that resembles the behavior of a large complex model. The large complex model is first trained with large datasets. Then, output predictions produced from the larger one is transferred to a smaller DNN model at which functions learned from the large model is approximated. The authors in [41] developed a multi-teacher distillation framework for compressed video action recognition based on convolutional neural networks. They showed how model is compressed by transferring a weighted average of multiple teachers' knowledge to a single student. The study in [42] proposed to only train the backbone of a student model to mimic the feature extraction output of the teacher model. The authors argue that to first distil the backbone knowledge from teacher and then to fit the task-head with labeled data can improve the generalization ability of knowledge distillation method.

3.2 Distributed Training and Inference Tasks of Deep Learning

The multilayer structure of deep learning is more suitable for edge computing. Specifically, when there are compute-intensive tasks like huge DNN, multiple distributive edge nodes, cloud datacenters and IoT end devices can work collaboratively to accomplish the task to meet the requirements of the intelligent system [43, 44]. For instance, in the case of cloud-edge collaboration, models are usually trained on the cloud and then transferred to the edge for inference tasks. In such type of scenarios, we need to focus on the distributed DNN models over the cloud and edge edge to realize real-time DNA-based intelligent services and applications.

Numerous frameworks have been provided reliable and efficient solutions for optimizing edge computing implementations. The authors in [45] designed a framework called 'In-Edge AI' to intelligently utilize the collaboration of end devices and edge nodes to exchange the learning parameters for a better training and inference of the models, They integrate deep reinforcement learning techniques and federated learning framework with mobile edge systems to optimize mobile edge computing, caching and reduce the unnecessary system communication load. The study in [46] presented an architecture that

is based on a distributed edge and cloud paradigm. It provides a balance between the benefits and cost of data processing at the edge versus at the cloud centric computing. The flexible framework proposed in [47] combines deep learning in IoT and flexible edge computing architecture using multiple agents to significantly improve performance of the system. The proposed model optimizes the task assignment between the edge and cloud layers. The authors in [48] introduces deep learning for IoTs into the edge computing environment and provides the deep learning layer service to manage the task at the edge computing to improve the performance of the IoT deep learning applications. There are different architectures and methods to speed up deep learning training and inference on the edge.

Table 1. Training and Inference Architectures of Deep Learning on the Edge

DNN Architectures of edge computing	
Training Architectures	Inference Architectures
Centralized Decentralized Hybrid	Edge based Device based Edge-Device based Edge-Cloud based Device-Edge-Cloud based

1) Training Architectures of DNN

The three main training architecture modes of DNN are: Centralized, Decentralized, and Hybrid.

a) Centralized Training Architecture

In the centralized training architecture mode, the DNN model is trained in the cloud datacenter after gathering data from IoT end devices [22,49]. The IoT end devices include smart phones, raspberry pi, medical devices, smart cars or surveillance cameras, etc. First, the training data originated from distributed IoT end devices is sent to the cloud data center. Then, the cloud datacenter performs the DNN training using these IoT data (See Fig. 4). After the DNN model is trained, it may be deployed back on the IoT end devices for serving the user. One of the drawbacks of centralized training architecture is that the data originated on the IoT end devices may be sensitive to send it to the cloud datacenter. The other drawback is transferring massive IoT big data may be costly and slow, especially if the deep learning model needs to be updated frequently.

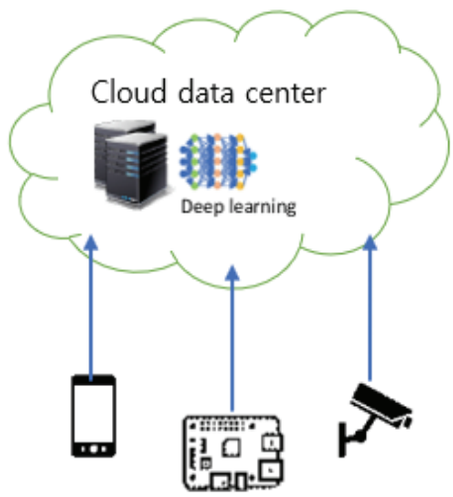

Fig. 4. Centralized Training Architecture

b) Decentralized Training Architecture

In the decentralized training architecture mode, each IoT end device trains its own DNN model locally with their own local data keeping the data on the device themselves (See Fig. 5). This allows to preserve private information locally as it eliminates copying of data to the cloud datacenter. The global DNN model can be obtained by sharing local training improvements [22, 49]. To do that, IoT end devices in the network communicate with each other to exchange the local model updates with distributed edge training methodologies, such as federated learning and large batch training.

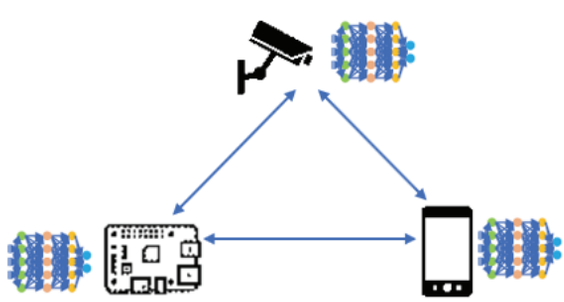

Fig. 5. Decentralized Training Architecture

c) Hybrid Training Architecture

The hybrid training architecture mode combines both the centralized and decentralized training architectures [22, 49]. The DNN model is trained on either a cloud datacenter or by edge servers receiving decentralized updates from each IoT devices (See Fig. 6). This architecture is also known as Cloud-Edge-Device training due to the involved roles of end device, edge devices and cloud datacenters.

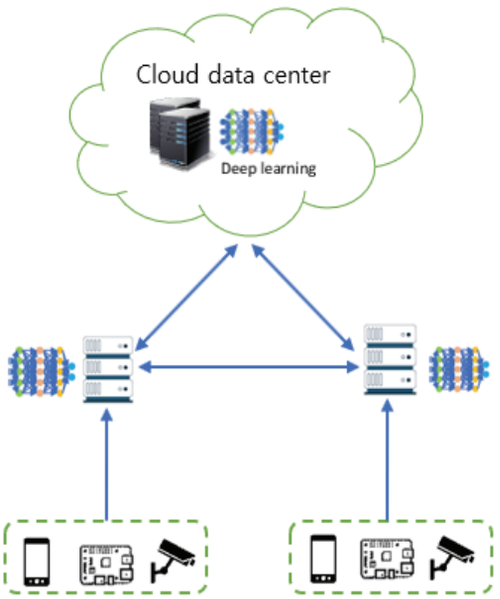

Fig. 6. Hybrid Training Architecture

2) Inference Architectures of DNN

Inference refers to the process of using a trained DNN model to provide the best possible accuracy. It requires a pre-trained DNN model to infer a result. When a new IoT raw data is given as input to the trained DNN model, it outputs a prediction based on accuracy of the trained DNN model. As IoT end devices are compute and memory-constrained, they cannot fully execute the DNN model. One possible solution is to distribute the DNN model across the three layers of edge computing. An architecture of the DNN model inference is crucial to have high-quality deep learning service in edge computing. Several major edge centric inference architectures are defined in the literature [29, 49-51]. For instance, in [49], the major edge centric inference modes are classified into four modes, namely edge-based, device-based, edge-device and edge-cloud.

a) Edge-based Mode

In this approach, the IoT end device receives the input data then send them to the edge server for inference computation (See Fig. 7). After inference is performed by the DNN model in the edge server, the prediction results will be returned to the IoT end device. In the Edge-based mode, it is easy implement the application on different mobile platforms as the DNN model is on the edge server, but the main disadvantage is that the inference performance depends on network bandwidth between the device and the edge server.

b) Device-based Mode

In this approach, the IoT end device acquires the DNN model from the edge server and accomplishes the inference locally in the IoT end device (See Fig. 8). The IoT end device does not communicate with the edge server during the process of Inference. So, the inference is reliable, but the IoT device requires huge resources such as CPU, GPU, RAM.

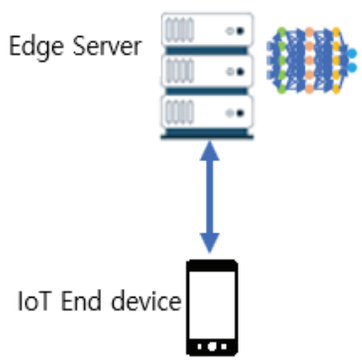

Fig. 7. Edge-based Inference Mode

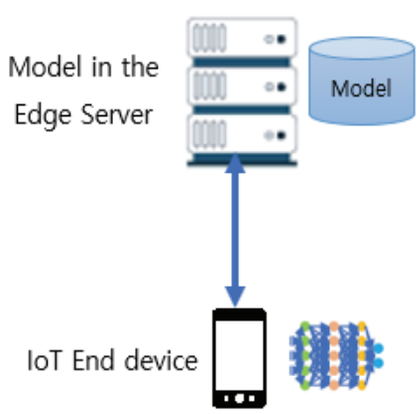

Fig. 8. Device-based Inference Mode

c) In Edge-Device Mode

In this mode, the IoT end device partitioned the DNN model into multiple layers and execute the first specific layers of the DNN model and send the intermediate data to the edge server where the remaining layers of the DNN model are executed (See Fig. 9). Then the edge server sends the prediction results to the IoT end device. Edge-device mode is more reliable and flexible than the edge-based and the device-based modes. However, the IoT end device in this mode requires large amount of resources as the first specific layers of the DNN model is computationally intensive.

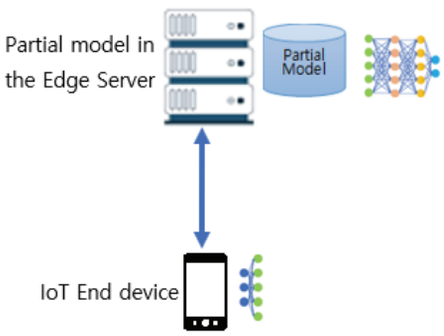

Fig. 9. Edge-device Inference Mode

d) The Edge-Cloud Mode

In this mode, the IoT end device is responsible for gathering input data. Parts of the first specific layers of the DNN model are executed in the edge server, and intermediate results are sent to the cloud datacenter (See Fig. 10). Then, the remaining layers of the DNN model are executed in the cloud datacenter. This mode is suitable for the case that the IoT end device is highly resource constrained.

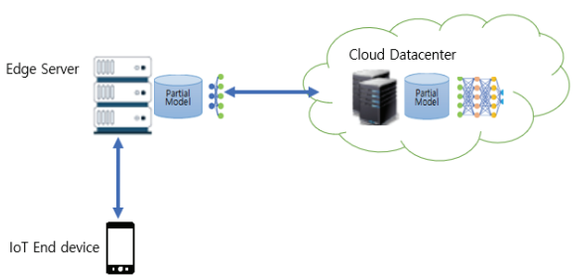

Fig. 10. Edge-cloud Inference Mode

e) The Device-Edge-Cloud Mode

Additional inference modes other than the 'Major edge-centric inference modes' can be considered to carry out complex DNN model inference tasks. For instance, the 'cloud-edge-device' inference mode can be assumed to increase reliability and flexibility by efficiently considering the heterogeneous resources across IoT end devices, edge servers and cloud datacenters. In cloud-edge-device mode, the IoT end device first partitions the DNN model into multiple layers based on the current system environmental factors such as device resource, network bandwidth, and edge server workload. It then executes the DNN model up to a specific layer and send the intermediate data to the edge server. Then, the edge server executes the next portion of layers and send the intermediate data to the cloud datacenter. Finally, the cloud data center

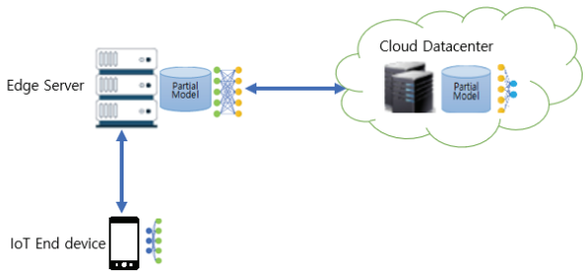

Fig. 11. Cloud-edge-device Inference Mode

executes the remain layers and send the prediction results to the IoT end device (See Fig. 11).

4. Distributed Training for Deep Learning

As DNN grows in complexity and datasets increase in size, the required processing and memory demands of deep learning increases, and a cluster of devices with high-performance that work together is required. Therefore, distributed, and parallel algorithms are vital for DNN to considerably reduce training times and make it suitable for real-time DNA-based intelligent services and applications. Distributed heterogeneous nodes work in parallel to speed up the DNN training due to the increased computational requirements of training DNN models and massive training datasets [52,53]. The training is carried out in multiple distributed nodes that are equipped with multiple CPUs or GPUs. The workload of compute and time-intensive tasks to train the DNN is split up and shared among multiple computer nodes. There are two main types of distributed DNN training: Data parallelism and Model parallelism [54]

4.1 Data Parallelism

In data parallelism based distributed DNN training, an identical replica of the DNN model is loaded in each worker station (node). The training data is split into non-overlapping partitions, where the number of partitions is equal to the total number of available nodes. Then, each worker node processes a subset of the training data. Data parallelism is more suitable when training data is too large to be stored on a single node or faster training is a requirement to provide real-time DNA-based intelligent services and applications. Data parallelism requires updates of parameters of the model as data is distributed on multiple worker nodes. Hence, the parameters of the model among nodes need to be synchronized at the end of an iteration in the parameter server (See Fig. 12). Data parallelism is

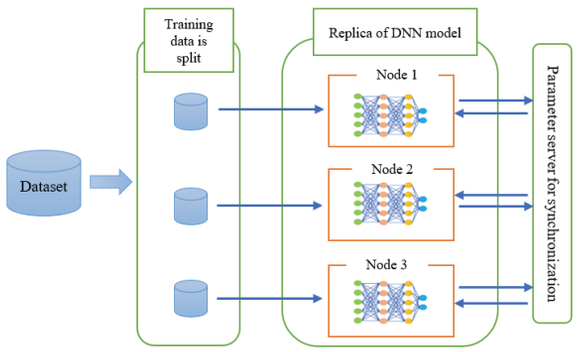

Fig. 12. Data Parallelism

efficient for compute intensive DNN models that have only a few parameters, such as the convolutional neural networks. But it does not scale well for DNN models that have large number of parameters as synchronization of the parameters becomes the bottleneck [55].

4.2 Model Parallelism

In model parallelism based distributed DNN training, the DNN model is segmented into disjoint subsets of a neural network that can run concurrently across multiple dedicated worker nodes (See Fig. 13) [56]. Each worker node runs on the same replica of the training data and eliminates parameter synchronization among worker nodes. But model parallelism requires data transfers between operations and prohibits parallelism within an operation. Model parallelism is more suitable if the model is too big to be fit into a single worker node. The main advantage of model parallelism is the low memory demand expected from each worker node as the model is split into disjoint subsets of the DNN across multiple worker nodes. Its disadvantage is the heavy communication that is needed among worker nodes and synchronization delays. In the model parallelism approach, worker nodes only need to synchronize the shared parameters, usually once for each forward or backward-propagation step.

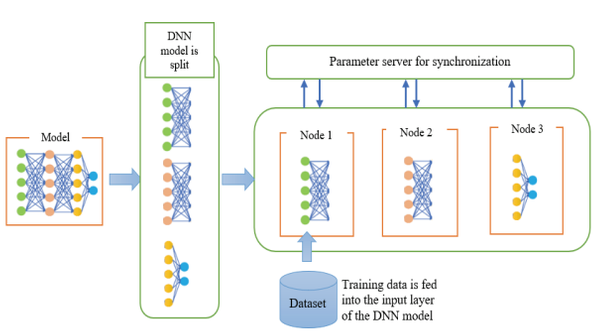

Fig. 13. Model Parallelism

4.3 Pipeline Parallelism

This approach is a combination of data and model parallelism. It is an improvement to model parallelism, where several mini batches are loaded into the system at once to effectively use resources parallelly [88]. First, the DNN training model is split into partitions and each partition is loaded to every worker node. The training data is also split into mini batches. Then, each worker node in the forward pass compute the output signal for a set of mini batches and immediately transmit them to the subsequent worker nodes. Similarly, worker nodes in the backpropagation pass compute the gradients for their model partition for multiple mini batches and immediately propagate them to the preceding worker nodes. Compared to pure model parallelism, pipeline parallelism significantly increases the utilization of worker nodes. This is because in the case of pure model parallelism only one batch is processed at a time.

5. Privacy Preserving Deep Learning in the Edge Computing

All efforts to enhance the performance of DNA-based intelligent services and applications in edge computing should be built with privacy in mind. IoT end devices would generate a massive volume of IoT data such as user location data, health, or records of activities at the network edge. The traverse of such data from the IoT end devices to the edge servers may contain sensitive information that may lead to privacy concerns. The training with distributed devices raises a privacy issue as there is a share of sensitive information associated with private data.

To realize DNA-based intelligent services and applications on IoT big data in distributed edge computing, heterogeneous IoT end devices and edge servers are required to work cooperatively. In that case, training or inference tasks might be sent to unfamiliar devices for further processing. Therefore, decentralized trust is required to have trustworthy deep learning in edge computing that are processed by different entities. Lightweight and distributed security designs are critical to ensure user authentication, access control and data integrity, and mutual platform verification for deep learning in edge computing. In addition to that, it is necessary to innovate secure routing schemes and trust network topologies for edge computing to deliver quality smart services as there is a case of coexistence of trusted edge nodes with malicious ones.

There are different approaches, such as cryptographic and/ or adding of noise to training data/ transmitting gradients, transmitted only partial data, etc., to improve privacy:

5.1 Cryptography Techniques

If the data is not encrypted, there could be risk of privacy leakage and attacks from malicious users. Some works such as [57, 58] suggest encryption of data to protect privacy. However, the encrypted data need to be decrypted before the execution of training or inference tasks. This process requires additional computation overhead. To cope with this problem, future efforts could pay more attention to cognitive homomorphic encryption method [59], as this encryption method allows direct computation on ciphertexts and generate encrypted results. After decryption, the result is the same as the result achieved by computation on the unencrypted data. The authors in [60] propose multi-key privacy preserving deep learning based on a hybrid structure by combining the double decryption mechanism [61] and fully homomorphic encryption [62] to avoid the interaction among multiple data owners. . Suggested work in [89] proposes a CryptoNN framework that trains a neural network using encrypted data without communication protocol overheads. The CryptoNN scheme simply computes permitted functions over sensitive data encrypted by distributed data owners. It only acquires computed results instead of the plaintext. The framework supports predictive analysis in privacy-preserving way.

5.2 Adding Noise to the Training Data

While it is prevalent to outsource training of deep learning models in the edge computing, protecting the privacy of sensitive training data and preventing information leakage to untrusted edge nodes is very essential. The authors in [63] propose a privacy preserving through data obfuscation to preserve the privacy of training data in machine learning applications. The obfuscate function that they introduced is applied to the training data before training. The obfuscate function adds random noise to existing data to hide sensitive information from untrusted third parties.

5.3 Adding Noise to the Transmitted Gradients

One of the approaches to protect privacy in relation with training a DNN is to add noise to the transmitted gradients [64, 65]. Based on the authors in [64], some gradients above a certain given threshold are selected to be uploaded to the central server, and noise is added to each of the uploaded gradient to train the model accurately while reducing information leakages from the training updates. A similar problem studied in [65] modified the gradient by clipping, averaging, and adding noise to it before uploading to the central server.

5.4 Transmitting only Partial Data

This is achieved by splitting the DNN model across the IoT end device, the edge devices and cloud datacenters. For instance, the author in [66] design a cloud-based framework that partitions the DNN model across the end devices and cloud datacenters. A lightweight privacy-preserving mechanism that consists of an arbitrary data nullification and addition of random noise is introduced to guarantee privacy. Data transformation is performed on the end device, while the training and inference tasks rely on the cloud datacenter. In the study presented in [67], model partitioning is combined with differential privacy to enable the privacy preserving edge-based training of DNN models. The DNN model is split between the end devices and the edge server in a way that both data and parameters are protected. The initial layers of the DNN model are computed on the end device and are mixed with noise before they are being uploaded to the edge server to obfuscate the training data and preserve privacy from untrusted edge servers.

6. Applications of Deep Learning in the Edge Computing

Deep learning requires high computational resources which may not be available at the IoT end devices. Edge computing, where distributed computing devices are placed near to the IoT end devices, is a possible solution to meet the high computational resources and low latency requirements of deep learning. Therefore, moving deep learning into edge computing helps to unleash the potentials of IoT big data to satisfy the ever-expanding demands of various DNA-based intelligent services and applications. In this section, we conduct an overview of research works on the different application domains where deep learning on the network edge platform can be useful.

6.1 Adding Natural Language Processing (NLP)

Deep learning requires high computational resources which may not be available at the IoT end devices. Edge computing, where distributed computing devices are placed near to the IoT end devices, is a possible solution to meet the high computational resources and low latency requirements of deep learning. Therefore, moving deep learning into edge computing helps to unleash the potentials of IoT big data to satisfy the ever-expanding demands of various DNA-based intelligent services and applications. In this section, we conduct an overview of research works on the different application domains where deep learning on the network edge platform can be useful.

6.2 Image and Video

Deep learning has seen astonishing success for complex tasks such as object detection [73] and image classification [74] problems that can be used for image and video analysis. For instance, camera frames of an autonomous vehicle require real-time inferences to detect and avoid obstacles. The delay to access cloud services might not be short enough to satisfy the requirements of time-critical applications like cooperative autonomous driving [5]. A viable approach is to move applications of deep learning, such as camera frames of an autonomous vehicle, surveillance videos, etc., towards the network edge to satisfy the latency requirements. Some studies also integrate edge and cloud computing to guarantee high accuracy with low latency considering the computational limitations of edge platforms [75, 76].

6.3 Smart City

Smart cities use IoT big data generated from IoT end devices and DNN to extract insights to efficiently manage city resources and services, such as monitoring traffic and transportation systems, water supply networks, crime detection , power plants, and many aspects of livings across the city [77-79]. In traffic flow management, data collected from different roads can be analyzed with deep learning to predict traffic flows, road closures and suggest alternative roads [80, 81]. Considering the resource limitations of IoT end devices, edge computing is a viable solution for the computation intensive DNA based smart city applications. For instance, the integration of edge computing and deep learning enables a city to provide efficient energy management in smart cities [82].

6.4 Smart Healthcare

DNA-based Healthcare is progressively leveraging IoT for delivering smart healthcare to speed up health diagnostics and reveal critical conditions by using deep learning. IoT health related data can be captured from IoT devices, such as wearable, ingestible and embedded sensors, etc., and processed by using deep Learning based approaches. Edge computing is a promising solution that pushes computing intensive tasks, such as healthcare services from the IoT end devices to the network edge platform [83]. For instance, the study in [84] integrates deep learning and edge computing devices to provide a real-life application of heart disease analysis by training neural networks on popular datasets Edge-cognitive-computing-based smart-healthcare system addresses emergency situations of patients and to provide personalized healthcare services for special users [85]. Information extracted from an image the help of DNN helps to detect different types of cancers, such as, skin cancer [86] and breast cancer [87].

7. Open Issues and Challenges

Although the convergence of deep learning and edge computing has opened many opportunities, there are also some challenges which constrained its enhancement. The following section addresses some of the open issues and challenges.

7.1 Trade-offs Among the Various Performance Indicators of the Training and Inferring Architectures

For DNA-based intelligent services with a specific mission, there is usually a series of deep learning model candidates in the edge computing that can accomplish the mission of the DNA-based intelligent services depending on training and inferring architectures of deep learning and various performance indicators. However, it is difficult for developers to choose an appropriate model for DNA-based intelligent services as the standard performance indicators fail to reflect the runtime performance of training and inference models of the edge intelligence. For instance, to better assess a distributed training method, the performance indicators, such as training loss, convergence, privacy, communication cost, latency, and energy efficiency can be utilized. Likewise, the key performance indicators such as latency, accuracy, energy, and communication overhead can be considered to evaluate

the service quality of the DNA-based intelligent services inference model. Therefore, to choose the best and appropriate deep learning model that can be used with edge computing, it is necessary to explore the trade-offs among the various performance indicators of the training and inferring architectures of deep learning.

7.2 Design of Computation-aware Networking Technologies for Efficient Communication

Training the DNN model across distributed nodes of the edge computing platform is data intensive and incurs communication overhead as raw data or intermediate data is transferred across the computing nodes. The communication overhead increases the training latency and bandwidth consumption. Therefore, advanced computation- aware network technologies are required to efficiently share raw data or intermediate data (computation results) of deep learning applications across the distributed edge nodes of edge computing platform. The emerging 5G communication technology together with advanced techniques such as Software-Defined Network (SDN) and Network Function Virtualization (NFV) could bring new possibilities towards a feasible solution to efficiently share computation results of deep learning across the distributed nodes of edge computing platform. The SDN and NFV allow flexible control over the network resources and support on demand interconnections across the various edge nodes for training and inferencing of computation intensive DNN models. Despite of the preliminary foundation of feasible network solutions, there is still a long way to go to address the communication overhead and to realize DNA-based intelligent services and applications in the edge computing platform.

7.3 System Integration, Verification, and Testing of DNA-based Intelligent Applications

To decide whether a DNN model applies to a given DNA-based intelligent application on the network edge, performing system integration among the IoT big data analysis and distributed nodes in the edge computing platform is vital. On one hand, the integration of IoT big data that put great demands to solve large-scale complex problems comes with its share of challenges. The underlying heterogeneity in individual data, and the large size of datasets lead to compute intensive analysis. Lack of studies in prioritizing the diverse set of tools makes heterogeneous IoT big data integration and analysis a challenging task. An integrative analysis of heterogeneous IoT big data that aims to ease the interoperability of multiple datasets is required. Hence, a framework that can help in a seamless analysis of heterogeneous IoT big data needs to be established. On the other hand, distributed edge computing is a relative concept that refers to computing, storage, and network resources between data sources and cloud computing center paths. The network edge resources mainly include terminals such as computers, Wi-Fi access points, cellular network base stations routers, and other infrastructure. These resources are numerous, independent, and scattered around users. It is required to integrate these independent and decentralized resources to provide computing, storage, and network services for DNA-based intelligent applications on the network edge. Leveraging deep learning within edge computing should meet the key requirements of industry digitization in agile connection, low data latency for efficient and real-time DNA-based intelligent services and applications, security and privacy protection. In this context, verification and test need to be carried out based on the key requirements. The obtained results must be analyzed and compared to the expected results of DNA-based intelligent services and applications. Verification strategies that include verification scope and verification techniques need to be established. verification scope can be identified by listing as many characteristics as possible. For instance, it may include latency threshold for efficient and real-time smart services, security, and privacy protection etc. Verification techniques can include inspection, analysis, simulation, peer-review, testing, etc., to perform verification actions according to a given constraints. Integration testing of a component and module testing of a system (as a whole) helps to verify whether the smart system works as expected or not.

8. Conclusion

In this paper, we conducted a thorough review of the current state of leveraging deep learning in edge computing to facilitate computation-intensive DNA-based intelligent services in resource-constrained IoT environments. Edge computing, where distributed computing devices are placed near to the IoT end devices, is a possible solution to unleash the potential of IoT big data generated from IoT end devices. We first introduce the basic concepts to highlight the key advantage of running deep learning at the network edge platform. Specifically, we present the

three most widely applied DNN structures: MLPs, CNN, and RNNs, and provide the two main improvements that speed-up deep learning in the edge computing platform. The first enhancement is to design an efficient deep learning model that reduces computational and memory space requirements to train the model with less time. The second one is to distribute the training and inference tasks of deep learning among the IoT end devices, the edge data servers, and the cloud datacenters for parallel processing to obtain better efficiency. The two main types of distributed DNN training - Data parallelism and Model parallelism - are also presented as distributed and parallel algorithms are vital for DNN to considerably reduce training times and make it suitable for DNA-based intelligent services and applications. We then provide the different privacy-preserving approaches, such as cryptographic and adding noise techniques, of deep learning on the edge computing environment to protect sensitive information that may lead to privacy concerns. Then, various application domains, such as NLP, Image and video, smart city, and smart healthcare, where deep learning on the network edge platform can be useful are also presented. Finally, we highlight open issues and challenges leveraging deep learning within the edge computing environment. We believe that this overview enables to understand the recent trends and opportunities of the various approaches that speed up the execution of training and inference of deep learning across distributed nodes of the edge computing environment.

References

[1] International Data Corporation (IDC) [Internet], https://www.idc.com/getdoc.jsp?containerId=prUS45213219#:~:text=A%20new%20forecast%20from%20International,these%20devices%20will%20also%20grow.

[2] L. R. Zheng, H. Tenhunen, and Z. Zou, "Smart Electronic Systems: Heterogeneous Integration of Silicon and Printed Electronics," *John Wiley & Sons*, 2018.

[3] M. Anandhalli and V. P Baligar, "A novel approach in real-time vehicle detection and tracking using Raspberry Pi," *Alexandria Engineering Journal*, Vol.57, Issue 3, pp.1597-1607, 2018.

[4]. M. Syafrudin, G. Alfian, N. L. Fitriyani, and J. Rhee, "Performance Analysis of IoT-Based Sensor, Big Data Processing, and Machine Learning Model for Real-Time Monitoring System in Automotive Manufacturing," *Sensors*, Vol.18, Issue 9, pp.2946, 2018.

[5] H. Khelifi, S. Luo, B. Nour et al., "Bringing deep learning at the edge of information-centric internet of things," *IEEE Communications Letters*, Vol.23, Issue 1, pp.52-55, 2019.

[6] Y. Liu, C. Yang, L. Jiang, and Y. Zhang, "Intelligent Edge Computing for IoT-Based Energy Management in Smart Cities," *IEEE Network*, Vol.33, Issue 2, pp.111-117, 2019.

[7] M. Merenda, F. G. Praticò, R. Fedele, R. Carotenuto, and F. G. D. "A Real-Time Decision Platform for the Management of Structures and Infrastructures," *Electronics,* Vol.8, Issue 10, pp.1180, 2019.

[8] S. E. Bibri, "The IoT for smart sustainable cities of the future: An analytical framework for sensor-based big data applications for environmental sustainability," *Sustainable Cities and Society*, Vol.38, pp.230-253, April, 2018.

[9] C. Ieracitano, N. Mammone, A. Hussain and F. C. Morabito, "A novel multi-modal machine learning based approach for automatic classification of EEG recordings in dementia," *Neural Networks*, Vol.123, pp.176-190, March, 2020.

[10] A. Rajkomar, J. Dean, and I. Kohane, "Machine learning in medicine," *The New England Journal of Medicine*, Vol.380, pp.1347-1358, 2019.

[11] K. Y. Ngiam and I. W. Khor, "Big data and machine learning algorithms for health-care delivery," *The Lancet Oncology*, Vol.20, Issue 5, pp.e262-e273, 2019.

[12] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, Vol.61, pp.85-117, Jan. 2015.

[13]. B. Arshad, R. Ogie, J. Barthelemy, B. Pradhan, N. Verstaevel, and P. Perez, "Computer Vision and IoT-Based Sensors in Flood Monitoring and Mapping: A Systematic Review," *Sensors*, Vol.19, Issue 22, pp.5012, 2019.

[14] B. Ravandi and I. Papapanagiotou, "A self-learning scheduling in cloud software defined block storage," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp.415-422, Jun. 2017.

[15] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," in *Proc. Of ACM SIGCOMM*, 2015.

[16] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," *IEEE Internet of Things Journal*, Vol.6, Issue 2, pp.3512-3524, 2019.

[17] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, "An edge-computing based architecture for mobile augmented reality," *IEEE Network*, Vol.33, Issue 4, pp.162-169, 2019.

[18] M. Satyanarayanan, "The emergence of edge computing," *Computer (Long. Beach. Calif).*, Vol.50, Issue 1, pp.30-39, 2017.

[19] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, Vol.19, Issue 4, pp.2322-2358, 2017.

[20] S. Weisong, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal,* Vol.3, Issue 5, pp.637-646, 2016.

[21] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning," *Nature 521*, Vol.7553, pp.436-444, 2015

[22] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, "Deep Learning," Vol.1. Cambridge, MA, USA: MIT Press, 2016.

[23] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A Survey on Network Methodologies for Real-Time Analytics of Massive IoT Data and Open Research Issues," *IEEE Communications Surveys and Tutorials,* Vol.19, Issue 3, pp.1457-1477, 2017.

[24] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research,* 12 ARTICLE, pp.2493-2537, 2011.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM,* Vol.60, Issue 6, pp.84-90, 2017.

[26] M. McClellan, C. Cervelló-Pastor, and S. Sallent, "Deep Learning at the Mobile Edge: Opportunities for 5G Networks," *Applied Sciences,* Vol.10, Issue 4, pp.4735, 2020.

[27] A. L. Caterini and D. E. Chang, "Recurrent neural networks," *Design and Applications,* Vol.5, 2018.

[28] P. Agarwal, and M. A. Alam, "Lightweight Deep Learning Model for Human Activity Recognition on Edge Devices," *Procedia Computer Science,* Vol.167, 2020

[29] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-Demand Accelerating Deep Neural Network Inference," *IEEE Transactions on Wireless Communications,* Vol.19, Issue 1, 2019

[30] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp.2204-2713, 2018.

[31] W. Meng et al., "Two-bit networks for deep learning on resource-constrained embedded devices," *arXiv preprint arXiv:1701.00485,* 2017.

[32] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-oriented Approximation of Convolutional Neural Networks," *arXiv preprint arXiv:1604.03168,* 2016.

[33] Intel Developer Zone [Internet], https://software.intel.com

[34] C. Yang, Z. Yang, A. M. Khattak, L. Yang, W. Zhang, W. Gao, and M. Wang, "Structured Pruning of Convolutional Neural Networks via L1 Regularization," *IEEE Access,* Vol.7, pp.106385-106394, 2019.

[35] F. Tung and G. Mori, "Deep Neural Network Compression by In-Parallel Pruning-Quantization," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2020.

[36] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv: 1607.03250,* 2016.

[37] J.-H. Luo, J. Wu, and W. Lin, "Thinet: a filter level pruning method for deep neural net- work compression," *Proceedings of the IEEE International Conference on Computer Vision,* 2017.

[38] S. Yao, Y, Zhao, A. Zhang, L. Su and T. Abdelzaher, "DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework," *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems,* pp.1-14, 2017.

[39] T. J. Yang, Y. H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp.5687-5695, 2017.

[40] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv preprint arXiv:1503.02531,* 2015.

[41] M.-C. Wu, C.-T. Chiu, and K.-H. Wu, "Multi-teacher knowledge distillation for compressed video action recognition on deep neural networks," *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE,* pp.2202-2206, 2019.

[42] M. Gao, Y. Shen, Q. Li, J. Yan, L. Wan, D. Lin, C. Change Loy, and X. Tang, "An embarrassingly simple approach for knowledge distillation," *arXiv preprint arXiv:1812.01819,* 2018.

[43] Y. Lin, S. W. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training," *arXiv preprint arXiv:1712.01887,* 2017.

[44] S. Teerapittayanon, B. McDanel and H.T. Kung, "Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices," *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, pp.328-339, 2017.

[45] 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 Network,* Vol.33, Issue 5, pp.156-165, 2019.

[46] S. B. Carlo, M. Touna, D. C. Verma, and A. Cullen, "Edge Computing Architecture for applying AI to IoT," *IEEE International Conference on Big Data (Big Data)*, IEEE, pp.3012-3016, 2017.

[47] S. Sureddy, K. Rashmi, R. Gayathri, and A. S. Nadhan, "Flexible Deep Learning in Edge Computing for Internet of Thinkgs," *International Journal of Pure and Applied Mathematics,* Vol.119, Issue 10, pp.531-543, 2018

[48] H. Li, K. Ota, and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," *IEEE Network,* Vol.32, Issue 1, pp.96-101, 2018.

[49] 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," *Proceedings of the IEEE,* Vol.107, Issue 8, pp.1738-1762, 2019.

[50] J. Chen and X. Ran, "Deep Learning with Edge Computing: A Review," P*roceedings of the IEEE,* Vol.107, Issue 8, pp. 1655-1674, 2019.

[51] M. Merenda, C. Porcaro, and D. Iero, "Edge Machine Learning for AI-Enabled IoT Devices: A Review," *Sensors.* Vol.20, Issue 9, pp.2533, 2020.

[52] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow, "A system for large-scale machine learning," *The 12th USENIX Conference on Operating Systems Design and Implementation({OSDI} 16),* 2016.

[53] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," *Advances in Neural Information Processing Systems,* Vol.25, pp1223-1231, 2012.

[54] R. Mayer and H.-A. Jacobsen, "Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques, and Tools," *ACM Computing Surveys (CSUR),* Vol.53, Issue 1, pp.1-37, 2020.

[55] Z. Jia, M. Zaharia, and A. Aiken, "Beyond data and model parallelism for deep neural networks," *arXiv preprint arXiv:1807.05358*, 2018.

[56] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," *Advances in Neural Information Processing Systems*, Vol.25, pp.1223-1231, 2012.

[57] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," arXiv preprint arXiv:1610.05492, 2016.

[58] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575,* 2015.

[59] C. Gentry, "Fully homomorphic encryption using ideal lattices," *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing,* pp.169-178, 2009.

[60] P. Li, J. Li, Z. Huang, T. Li, C.-Z Gao, S.-M. You, and K. Chen, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generation Computer Systems,* Vol.74, pp.76-85, 2017.

[61] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," *International Conference on the Theory and Application of Cryptology and Information Security,* Springer, Berlin, Heidelberg, pp.37-54, 2003.

[62] C. Gentry, "Fully homomorphic encryption using ideal lattices," *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing,* pp.169-178, 2009.

[63] T. Zhang, Z. He, and R. B. Lee, "Privacy-preserving machine learning through data obfuscation," *arXiv preprint arXiv:1807.01860,* 2018.

[64] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security,* pp.1310-1321, 2015.

[65] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep Learning with Differential Privacy," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security,* pp.308-318, 2016.

[66] 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," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining,* pp.2407-2416, 2018.

[67] Y. Mao, S. Yi, Q. Li, J. Feng, F. Xu, and S. Zhong, "Learning from differentially private neural activations with edge computing," *2018 IEEE/ACM Symposium on Edge Computing (SEC),* pp.90-102, 2018.

[68] D. Li and Y. Liu, "Deep Learning in Natural Language Processing," *Springer,* 2018.

[69] Amazon [Internet], "Alexa Voice Service," 2016.

[70] Team Siri, "Hey Siri: An On-Device DNN-Powered Voice Trigger for Apple's Personal Assistant," *Apple Machine Learning Journal,* Vol.1, Issue 6, 2017

[71] M. S. Mehrjardi, A. Trablesi, and O. R. Zaiane, "Self-Attentional Models Application in Task-Oriented Dialogue Generation Systems," *arXiv preprint arXiv:1909.05246,* 2019.

[72] S. P. Singh, A. Kumar, H. Darbari, L. Singh, A. Rastogi, and S. Jain, "Machine translation using deep learning: An overview," *2017 International Conference on Computer, Communications and Electronics (Comptelix),* pp.162-167, 2017.

[73] X. Jiang, A. Hadid, Y. Pang, E. Granger, and X. Feng, "Deep Learning in Object Detection and Recognition," *Springer:*, 2019.

[74] V. Aggarwal and G. Kaur, "A review:deep learning technique for image classification," *ACCENTS Transactions on Image Processing and Computer Vision,* Vol.4, Issue 11, pp.21-25, 2018.

[75] C. C. Hung, G. Ananthanarayanan, P. Bodík, L. Golubchik, M. Yu, V. Bahl, and M. Philipose, "VideoEdge: Processing Camera Streams using Hierarchical Clusters," *2018 IEEE/ACM Symposium on Edge Computing (SEC),* IEEE, pp.115-131, 2018.

[76] S. Mittal, N. Negi, and R. Chauhan, "Integration of edge computing with cloud computing," *2017 International Conference on Emerging Trends in Computing and Communication Technologies (ICETCCT),* pp.1-6, 2017.

[77] C. S. Lai, Y. Jia, Z. Dong, D. Wang, Y. Tao, Q. H. Lai, T. K. W. Richard, F. Z. Ahmed, R. Wu, and L. L Lai, "A Review of Technical Standards for Smart Cities," *Clean Technologies,* Vol.2, Issue 3, pp.290-310, 2020.

[78] J. Souza, A. Francisco, C. Piekarski, and G. Prado, "Data Mining and Machine Learning to Promote Smart Cities: A Systematic Review from 2000 to 2018," *Sustainability,* Vol.11, Issue 4, pp.1077, 2019.

[79] F. Cicirelli, A. Guerrieri, G. Spezzano, and A. Vinci, "An edge-based platform for dynamic smart city applications," *Future Generation Computer Systems,* Vol.76, pp.106-118, 2017.

[80] A. Essien, I. Petrounias, P. Sampaio et al., "A deep-learning model for urban traffic flow prediction with traffic events mined from twitter," *World Wide Web,* pp.1-24, 2020

[81] Z. Liu, Z. Li, K. Wu, and M. Li., "Urban traffic prediction from mobility data using deep learning," *IEEE Network,* Vol.32, Issue 4, pp.40-46, 2018.

[82] Y. Liu, C. Yang, L. Jiang, S. Xie, and Y. Zhang, "Intelligent edge computing for IoT-based energy management in smart cities," *IEEE Network,* Vol.33, Issue 2, pp.111-117, 2019

[83] L. Greco, G. Percannella, P. Ritrovato, and F. Tortorella, M. Vento, "Trends in IoT based solutions for health care: Moving AI to the edge," *Pattern Recognition Letters*, 2020.

[84] S. Tuli, N. Basumatary, S. S. Gill, M. Kahani, R. C. Arya, G. S.Wander, and R. Buyya, "Healthfog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heartdiseases in integrated iot and fog computing environments," *Future Generation Computer Systems,* Vol.104, pp.187-200, 2020

[85] M. Chen, W. Li, Y. Hao, Y. Qian, and I. Humar, "Edge cognitive computing based smart healthcare system," *Future Generation Computer Systems,* Vol.86, pp.403-411, 2018.

[86] A. Esteva, B. Kuprel, R. Novoa, J. Ko, S. Swetter, H. Blau and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *nature,* Vol.542, Issue 7639, pp.115-118, 2017.

[87] L. Tsochatzidis, L. Costaridou, and I. Pratikakis, "Deep Learning for Breast Cancer Diagnosis from Mammograms-A Comparative Study," *Journal of Imaging,* Vol.5, Issue 3, pp.37, 2019.

[88] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Computing Surveys (CSUR),* Vol.52, Issue 4, 2018.

[89] R. Xu, J. B. Joshi, and C. Li, "CryptoNN: Training neural networks over encrypted data," *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS).* IEEE, pp.1199-1209, 2019.

Temesgen Seyoum Alemayehu

https://orcid.org/0000-0001-7513-9021

e-mail : tomsymal@gmail.com

He received a Ph.D. degree in Computer Science and Engineering from Ajou University in 2018 and M.Sc. degree in Computer Science from Addis Ababa University, Ethiopia in 2008. He has been a research professor at Ajou University since 2020. His research interests are in the area of cyber physical systems, dependable systems, energy efficient communications and AI Data processing/algorithm design.

We-Duke Cho
https://orcid.org/0000-0002-2895-0791
e-mail : chowd@ajou.ac.kr

We-Duke Cho received the M.S. degree and the Ph.D. degree in electrical and electronic engineering from Korea Advanced Institute of Science and Technology (KAIST) in 1983 and 1987 Korea, respectively. He is currently professor department of Electronics Engineering College of Information Technology, director of UCRI(ubiquitous convergence research institute) and life-care science lab at Ajou University in Korea. And also he is director of Korea Association of Smart Home(KASH) and smart+interior Forum in Korea and adjunct professor at Stoney brook university in New York (SUNY) USA in 2010~2012, Vice President and PM of digital mobile communication of research division at KETI in 1990 ~2003, visiting researcher of personal mobile division at TTP/ Cambridge in U.K. in 1994 ~1995, TTP/Berkeley in USA in 1994 ~1995, and principal researcher of smart TV project at KAITECH in Korea in 1990 ~ 1991. He was project manager at DSP Lab of LGE in 1983~1990.

Research area: advanced digital signal processing, Artificial Intelligence of Thing (AIoT) device/data processing, smart care system design, Smart home/living space system design. Multi-sensory AI data analysis.