Edge computing-DL survey

聂聂

关注他

2 人赞同了该文章

题目：Convergence of Edge Computing and Deep Learning: A Comprehensive Survey

摘要：来自工厂和社区的无处不在的传感器和智能设备可保证大量数据，不断增加的计算能力正在推动计算和服务的核心，从云端到网络边缘。作为广泛改变人们生活的重要推动者，从人脸识别到雄心勃勃的智能工厂和城市，人工智能（尤其是深度学习）应用和服务经历了蓬勃发展的过程。然而，由于效率和延迟问题，当前的云计算服务架构阻碍了“为每个人和每个组织的每个组织提供人工智能”的愿景。因此，最近，更好的解决方案是从云到数据源附近的边缘释放深度学习服务。因此，旨在通过边缘计算促进深度学习服务部署的边缘智能受到了极大的关注。此外，深度学习作为人工智能的主要代表，可以集成到边缘计算框架中，为动态，自适应边缘维护和管理构建智能边缘。关于互利边缘智能和智能边缘，本文介绍和讨论：1）两者的应用场景; 2）实际实施方法和使能技术，即定制边缘计算框架中的深度学习训练和推理; 3）更普遍和细粒度智能的现有挑战和未来趋势。我们相信，这种调查可以帮助读者获取分散在通信，网络和深度学习中的信息，理解支持技术之间的联系，并促进关于边缘智能和智能边缘融合的进一步讨论。

关键词：边缘计算，终端云计算，计算卸载，人工智能，深度学习

I. INTRODUCTION

随着计算和存储设备的逐步普及，从云数据中心（云）中的服务器集群到个人计算机和智能手机，再到可穿戴设备和其他物联网设备，计算能力正在经历从云到的溢出边缘，例如500亿IoT（物联网）设备将在2020年之前连接到互联网[1]。另一方面，思科估计，到2021年，每年将在云外部生成近850个Zettabytes（ZB）数据，而全球数据中心流量仅为20.6 ZB [2]。这表明大数据的数据源也正在发生转变：从大规模云数据中心到越来越广泛的边缘设备。然而，现有的云计算逐渐无法管理和分析这些大规模分布式计算能力和数据：1）需要将大量计算任务传递到云进行处理[3]，这无疑对网络传输能力构成严峻挑战和云计算基础设施的计算能力; 2）具有新形式的各种应用具有严格的延迟要求，例如自动驾驶，而云无法提供与之匹配的快速响应，因为它远离用户[4]。

因此，出现了边缘计算[5]，[6]，并且希望计算任务尽可能接近数据源。 当然，边缘计算和云计算并不是相互排斥的[7]，[8]。 相反，边缘是云的补充或扩展。 与独立云计算相比，边缘计算与云计算相结合的优势主要体现在三个方面：1）骨干网缓解，分布式边缘计算节点可以处理大量计算任务而无需将所需数据传输到云端 从而缓解了网络的传输压力; 2）敏捷的业务响应，业务由边缘响应，消除数据传输的延迟，提高响应速度; 3）强大的云备份，云端可以提供强大的处理能力和边缘无法承受的海量存储。

作为最典型和更广泛使用的新形式的应用[9]，由于深度学习（DL）在计算机领域的巨大优势，各种基于深度学习的智能服务和应用已经改变了人们生活的方方面面。 视觉（CV）和自然语言处理（NLP）[10]。 这些成就不仅源于DL的发展，而且还与不断增长的数据和计算能力密不可分。 然而，对于更广泛的应用场景，例如智能城市，车辆互联网（IoV）等，由于以下因素，更广泛的智能服务仍然受到限制。

•成本：云中DL模型的培训和推理要求设备或用户将大量数据传输到云，从而消耗大量网络带宽;

延迟：使用云提供的服务可能会导致更高的传输延迟。 例如，自动驾驶汽车不能允许可能由云处理引起的毫秒延迟[11];

•可靠性：云计算完全依赖于无线通信和骨干网络，但对于工业制造场景（仅举几例），即使网络连接丢失，智能服务也必须高度可靠;

•隐私：DL所需的数据涉及大量私人信息，隐私问题对智能家居和城市等领域至关重要。

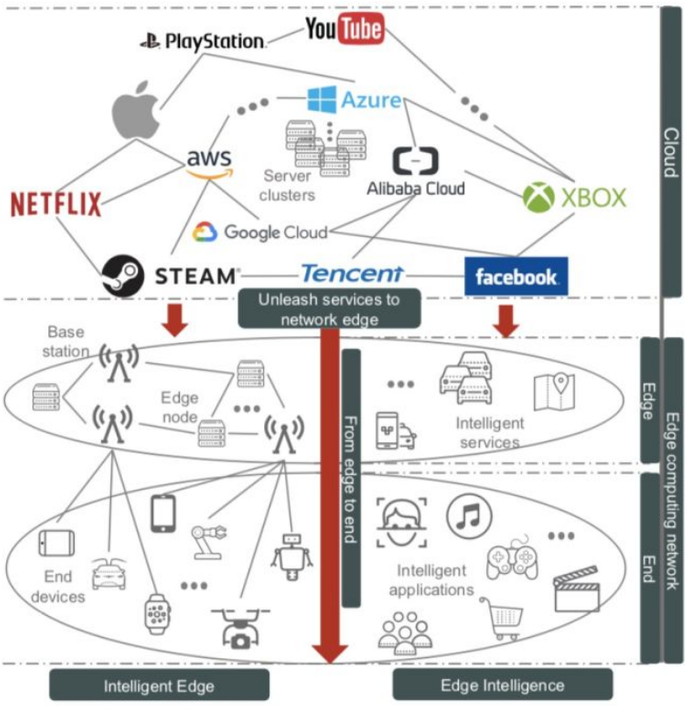

Fig. 1. Edge intelligence and intelligent edge.

由于边缘比云更接近用户，因此边缘计算有望解决这些问题。 事实上，边缘计算正在逐渐与人工智能（AI）相结合，在边缘智能和智能边缘的实现方面相互受益，如图1所示。边缘智能和智能边缘并不是彼此独立的。 边缘智能是目标，智能边缘的DL服务也是边缘智能的一部分。 反过来，智能边缘可以为边缘智能提供更高的服务吞吐量和资源利用率。

具体而言，边缘智能一方面希望尽可能地将DL计算从云推送到边缘，从而提供各种分布式，低延迟和可靠的智能服务。如图2所示，优点包括：1）DL服务部署在服务请求附近，云只在需要额外处理时才参与[12]，从而大大减少了向云发送数据的延迟和成本处理; 2）由于DL服务所需的原始数据本地存储在边缘或用户设备本身而不是云上，因此保护了隐私数据; 3）分层架构提供更可靠的DL计算; 4）通过更丰富的数据和应用场景，边缘计算可以促进DL的普遍应用，并实现“为每个人和每个组织提供AI的前景”[13]; 5）多样化和有价值的DL服务可以拓宽边缘计算的商业价值，加速其部署和增长。

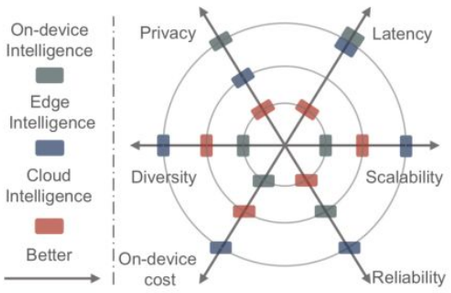

Fig. 2. Capabilities comparison of cloud, on-device and edge intelligence.

另一方面，智能边缘旨在将DL融入边缘，以实现动态，自适应边缘维护和管理。随着通信技术的发展，网络接入方法变得越来越多。同时，边缘计算基础设施充当中间媒介，使得无处不在的终端设备与云之间的连接更加可靠和持久[14]。此功能使终端设备，边缘和云逐渐成为共享资源的社区。然而，维护和管理如此庞大而复杂的整体架构（社区），包括但不限于无线通信，网络，计算，存储等，是一项重大挑战[15]。典型的网络优化方法依赖于固定的数学模型，但是，很难准确地模拟快速变化的边缘网络环境和系统。 DL有望解决这个问题：当面对复杂繁琐的网络信息时，DL可以依靠其强大的学习和推理能力从数据中提取有价值的信息并做出自适应决策，从而实现智能维护和管理。

因此，考虑到它们在多个方面都面临着一些相同的挑战和实际问题，如图3所示，我们考虑了构建边缘智能和智能边缘的五种支持技术：

1）DL on Edge（边缘DL应用），技术框架，用于系统地组织边缘计算和DL，以提供智能服务;

2）DL in Edge（边缘DL推理），侧重于边缘计算架构中DL的实际部署和推理，以满足不同的要求，如准确性和延迟;

3）Edge for DL（用于DL的边缘计算），其将边缘计算平台在网络架构，硬件和软件方面适应于用于DL计算的卡特;

4）DL at Edge（边缘DL训练），即在资源和隐私限制下训练分布式边缘设备边缘智能的DL模型;

5）DL for Edge（用于优化边缘的DL），用于维护和管理边缘计算网络（系统）的多个方面[16] [17]，例如边缘缓存，计算卸载，通信，网络，安全保护 等。

对于这些使能器，“DL on Edge”和“DL for Edge”分别与边缘智能和智能边缘的理论目标相对应，而“DL in Edge”，“Edge for DL”和“DL at Edge” 是他们的重要支持，涉及DL推理，边缘计算架构和DL培训。

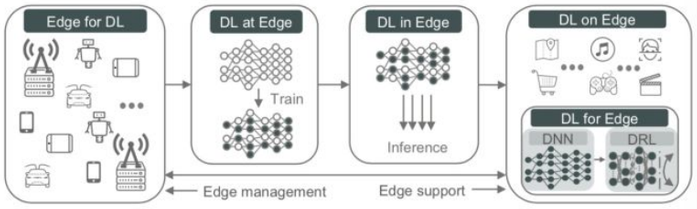

Fig. 3. Relationship between five enablers, i.e., DL on edge, DL in edge, DL at edge, edge for DL and DL for edge.

据我们所知，最相关的工作是[18]和[19]，但是，[18]涉及在无线通信角度促进边缘智能，即一方面利用边缘机器学习来改善通信，另一方面 另一方面，通过无线网络在网络边缘训练机器学习。 此外，关于DL推理和训练的讨论是[19]的主要贡献。 与[18]，[19]不同，本次调查在以下方面做出了贡献：1）通过边缘计算，跨网络，通信和计算全面考虑DL的部署问题； 2）从五个方面研究有关DL和边缘计算融合的整体技术范围； 3）指出DL和边缘计算是相辅相成的，**并且仅考虑在边缘部署DL是不完整的。**

本文的组织如下（如图4所示）。 我们在第一节中介绍了本次调查的背景和动机。然后，在第二节和第三节中分别给出了与边缘计算和DL相关的基础知识。 接下来，我们介绍五种启用技术，即边缘上的DL应用程序（第IV节），边缘上的DL推理（第V节），DL

服务的边缘计算（第VI节），边缘上的DL训练（第VII节）和DL上的DL训练 优化边缘（第八节）。 最后，我们在第IX节中讨论了现有的挑战和未来趋势，并在X节中总结了本文。所有相关的首字母缩写在表I中列出。

II. FUNDAMENTALS OF EDGE COMPUTING

边缘计算凭借其减少数据传输，改善服务等待时间和减轻云计算压力的优势，已成为突破新兴技术瓶颈的重要解决方案。 边缘计算架构将成为云的重要补充，甚至在某些情况下甚至可以替代云的角色。 更详细的信息可以在[8]，[20]，[21]中找到。

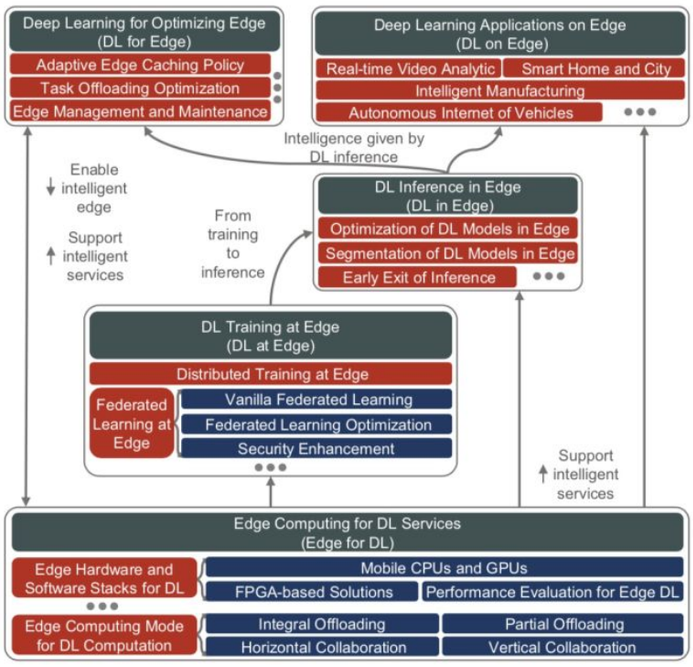

Fig. 4. Conceptual relationships of edge intelligence and intelligent edge.

Conceptual relationships of edge intelligence and intelligent edge.

TABLE I
LIST OF ABBREVIATIONS IN ALPHABETICAL ORDER

Abbr.	Definition	Abbr.	Definition	Abbr.	Definition
AC	Actor-Critic	DVFS	Dynamic Voltage and Frequency Scaling	NN	Neural Network
A3C	Asynchronous Advantage Actor-Critic	ECSP	Edge Computing Service Provider	NPU	Neural Processing Unit
AE	Auto-Encoder	EEoI	Early Exit of Inference	PPO	Proximate Policy Optimization
AI	Artificial Intelligence	EH	Energy Harvesting	QoE	Quality of Experience
APU	AI Processing Unit	ETSI	European Telecommunications Standards Institute	QoS	Quality of Service
AR	Augmented Reality	FAP	Fog radio Access Point	RNN	Recurrent Neural Network
ASIC	Application-Specific Integrated Circuit	FCNN	Fully Connected Neural Network	RoI	Region-of-Interest
BER	Bit Error Rate	FL	Federated Learning	RRH	Remote Radio Head
BS	Base Station	FPGA	Field Programmable Gate Array	RSU	Road-Side Unit
C-RAN	Cloud-Radio Access Networks	FTP	Fused Tile Partitioning	SDN	Software-Defined Network
CDN	Content Delivery Network	GAN	Generative Adversarial Network	SGD	Stochastic Gradient Descent
CNN	Convolutional Neural Network	GNNs	Graph Neural Networks	SINR	Signal-to-Interference-plus-Noise Ratio
CV	Computer Vision	IID	Independent and Identically Distributed	SNPE	Snapdragon Neural Processing Engine
D2D	Device-to-Device	IoT	Internet of Things	SRAM	Static Random Access Memory
DDoS	Distributed Denial of Service	IoV	Internet of Vehicles	TL	Transfer Learning
DDPG	Deep Deterministic Policy Gradient	KD	Knowledge Distillation	VM	Virtual Machine
DGC	Deep Gradient Compression	KNN	K-Nearest Neighbor	VR	Virtual Reality
DL	Deep Learning	MAB	Multi-Armed Bandit	V2V	Vehicle-to-Vehicle
DNNs	Deep Neural Networks	MDCs	Micro Data Centers	WLAN	Wireless Local Area Network
DQL	Deep Q-Learning	MDP	Markov Decision Process	ZB	Zettabytes
DRAM	Dynamic RAM	MLP	Multi-Layer Perceptron		
DRL	Deep Reinforcement Learning	NLP	Natural Language Processing		

A. Paradigms of Edge Computing(边缘计算范式)

在边缘计算的发展中，已经有各种针对网络边缘的新技术，它们具有相同的原理，但关注的重点不同，例如Cloudlet [22]，微数据中心（MDC）[23]，雾计算 [24] [25]和移动边缘计算[5]（即现在的多路访问边缘计算[26]）。 但是，边缘计算社区尚未就边缘计算的标准化定义，体系结构和协议达成共识[21]。 对于这套新兴技术，我们使用通用术语“边缘计算”。 在本节中，将介绍和区分沿时间线的不同边缘计算概念。

*1) Cloudlet and Micro Data Centers:*Cloudlets是结合了移动计算和云计算的网络架构元素。 它代表了三层架构的中间层，即移动设备，微云和云。 它的亮点在于努力：1）定义系统并创建支持低延迟边缘云计算的算法，以及2）在开源代码中实现相关功能，作为Open Stack云管理软件的扩展[22]。 与Cloudlets类似，MDC [23]也被设计用来补充云。 该想法是将运行客户应用程序所需的所有计算，存储和网络设备作为一个独立的安全计算环境封装在一个机箱中，以用于需要较低延迟的应用程序或电池寿命或计算能力有限的终端设备。

*2) Fog Computing:*雾计算的一大亮点是，它假设了一个具有数十亿个设备和大规模云数据中心的完全分布式的多层云计算架构[24] [25]。 尽管云和雾范式共享一组类似的服务，例如计算，存储和网络，但雾的部署却针对特定的地理区域。 此外，雾设计用于要求实时响应且延迟较少的应用程序，例如交互式和物联网应用程序。 与Cloudlet，MDC和MEC不同，雾计算更专注于IoT。

*3) Mobile (Multi-access) Edge Computing (MEC):*移动边缘计算将计算功能和服务环境置于蜂窝网络的边缘[5]。 它旨在提供更低的延迟，上下文和位置感知以及更高的带宽。 在蜂窝基站（BS）上部署边缘服务器使用户可以灵活，快速地部署新的应用程序和服务。 后来，ETSI（欧洲电信标准协会）通过适应更多的无线通信技术（例如Wi-Fi），将MEC的术语从移动边缘计算扩展到多路访问边缘计算[26]。

*4) Definition of Edge Computing Terminologies:*在大多数文献中，边缘设备的定义和划分是模棱两可的（边缘节点与终端设备之间的边界尚不清楚）。 因此，如图1所示，我们进一步将公共边缘设备分为终端设备和边缘节点：“终端设备”（终端级别）用于指代移动边缘设备（包括智能手机，智能车辆等）。）和各种IoT设备，“边缘节点”（edge level）代表“边缘节点”，包括但不限于Cloudlets，路边单元（RSU），Fog节点，边缘服务器，MEC服务器等， 即部署在网络边缘的服务器。

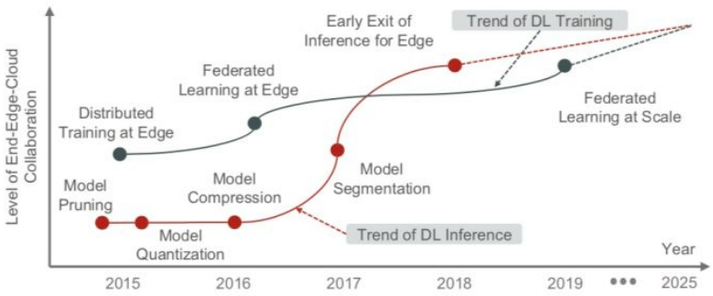

Fig. 5. Computation collaboration is becoming more important for DL with respect to both training and inference.

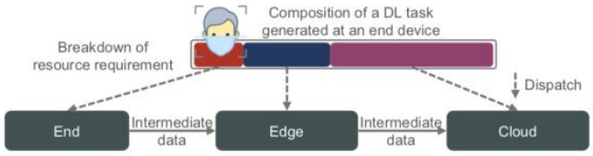

Fig. 6. A sketch of collaborative end-edge-cloud DL computing.

*5) Collaborative End-Edge-Cloud Computing*尽管云计算是为处理诸如DL之类的计算密集型任务而诞生的，但由于终端设备生成的数据量爆炸，它受到了限制。 因此，如图5所示，出现了用于DL的协作式边缘云计算。在这种新颖的计算范例中，如图6所示，可以直接在终端设备处执行具有较低计算强度的计算任务 端或被卸载到边缘，从而避免了由于将数据发送到云而造成的延迟。对于计算密集型任务，它将合理地分段并分别分发到末端，边缘和云以执行，从而在确保结果准确性的同时减少了任务的执行延迟[12]，[46]，[47] 。 这种协作范式的重点不仅是成功完成任务，而且还要实现设备能耗，服务器负载，传输和执行延迟的最佳平衡。

B. Edge Computing Hardware and Systems

TABLE II
SUMMARY OF EDGE COMPUTING AI HARDWARES AND SYSTEMS

	Owner	Production	Feature
Integrated Commodities	Microsoft	Data Box Edge [27]	Competitive in data preprocessing and data transmission
	Intel	Movidius Neural Compute Stick [28]	Prototype on any platform with plug-and-play simplicity
	NVIDIA	Jetson [29]	Easy-to-use platforms that runs in as little as 5 Watts
	Huawei	Atlas Series [30]	An all-scenario AI infrastructure solution that bridges "device, edge, and cloud"
AI Chips for Edge Devices	Qualcomm	Snapdragon 8 Series [31]	Powerful adaptability to major DL frameworks
	HiSilicon	Kirin 600/900 Series [32]	Independent NPU for DL computation
	HiSilicon	Ascend Series [33]	Full coverage from the ultimate low energy consumption scenario to high computing power scenario
	MediaTek	Helio P60 [34]	Simultaneous use of GPU and NPU to accelerate neural network computing
	NVIDIA	Turing GPUs [35]	Powerful capabilities and compatibility but with high energy consumption
	Google	TPU [36]	Stable in terms of performance and power consumption
	Intel	Xeon D-2100 [37]	Optimized for power- and space-constrained cloud-edge solutions
	Samsung	Exynos 9820 [38]	Mobile NPU for accelerating AI tasks
Edge AI Computing Systems	Huawei	KubeEdge [39]	Native support for edge-cloud collaboration
	Baidu	OpenEdge [40]	Computing framework shielding and application production simplification
	Microsoft	Azure IoT Edge [41]	Remotely edge management with zero-touch device provisioning
	Linux Foundation	EdgeX [42]	IoT edge across the industrial and enterprise use cases
	Linux Foundation	Akraino Edge Stack [43]	Integrated distributed cloud edge platform
	NVIDIA	NVIDIA EGX [44]	Real-time perception, understanding, and processing at the edge
	Amazon	AWS IoT Greengrass [45]	Tolerance to edge devices even with intermittent connectivity

在本节中，我们讨论了边缘智能的潜在启用硬件，即针对终端设备和边缘节点的定制AI芯片和商品。 此外，还介绍了用于DL的边缘云系统（表II中列出）。

*1) AI Chips for Edge Devices*新兴的边缘AI芯片根据其技术架构可以分为三类：1）基于GPU的芯片，它们倾向于具有良好的兼容性和性能，但通常会消耗更多的能量，例如基于Turing架构的NVIDIA GPU [35]；2）基于FPGA [48]，[49]的定制芯片，与GPU相比，它们节省能源并且需要较少的计算资源，但兼容性较差，编程能力有限； 3）ASIC（专用集成电路）芯片，例如Google的TPU [36]和海思的Ascend系列[33]，通常采用定制设计，在性能和功耗方面更稳定。

作为最繁荣的边缘设备，智能手机上的芯片发展迅速，其功能已扩展到AI计算的加速。 仅举几例，高通首先在Snapdragon中应用了AI硬件加速[31]，并发布了Snapdragon神经处理引擎（SNPE）SDK [50]，该工具支持几乎所有主要的DL框架。与高通相比，HiSilicon的600系列和900系列芯片[32]不依赖GPU。 但是，它还引入了神经处理单元（NPU）以实现向量和矩阵的快速计算，从而大大提高了DL的效率。 与HiSilicon和Qualcomm相比，联发科的Helio P60不仅使用GPU，而且还引入了AI处理单元（APU）来进一步加速神经网络计算[34]。 大多数商品芯片相对于DL的性能比较可以在[51]中找到，边缘设备的更多定制芯片将在后面详细讨论。

*2) Integrated Commodities Potentially for Edge Nodes*边缘节点应具有计算和缓存功能，并在终端设备附近提供高质量的网络连接和计算服务。 与大多数终端设备相比，边缘节点具有更强大的计算能力来处理任务。 另一方面，边缘节点可以比云更快地响应终端设备。 因此，通过部署边缘节点来执行计算任务，可以在确保准确性的同时加速任务处理。 此外，边缘节点还具有缓存功能，可以通过缓存受欢迎的内容来缩短响应时间。 例如，包括华为Atlas模块[30]和Microsoft的Data Box Edge [27]在内的实用解决方案可以进行初步的DL推断，然后转移到云中进行进一步的改进。

*3) Edge Computing Systems*边缘计算系统的解决方案正在蓬勃发展。 对于具有复杂配置和密集资源需求的DL服务，具有先进而卓越的微服务架构的边缘计算系统是未来的发展方向。 目前，Kubernetes是一种以容器为中心的主流系统，用于在云计算中部署，维护和扩展应用程序[52]。 华为基于Kubernetes，开发了其边缘计算解决方案“ KubeEdge” [39]，用于云与边缘之间的联网，应用程序部署和元数据同步（Akraino Edge Stack [43]也支持）。 “ OpenEdge” [40]专注于屏蔽计算框架并简化应用程序的生产。 对于物联网，Azure IoT Edge [41]和EdgeX [42]旨在通过在跨平台IoT设备上部署和运行AI将云智能传递到边缘。

III. FUNDAMENTALS OF DEEP LEARNING

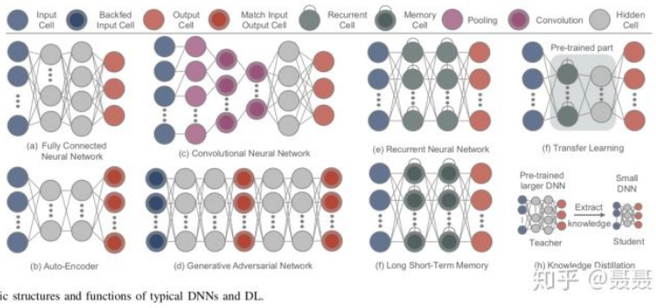

Fig. 7. Basic structures and functions of typical DNNs and DL.

关于CV，NLP和AI，DL已在众多应用中得到采用，并证实了其卓越的性能[65]。 当前，需要在云中部署大量GPU，TPU或FPGA来处理DL服务请求。 尽管如此，边缘计算体系结构由于覆盖了许多分布式边缘设备，因此可以用来更好地服务于DL。 当然，与云相比，边缘设备通常具有有限的计算能力或功耗。 因此，将DL与边缘计算结合起来并不容易，需要对DL模型和边缘计算功能进行全面的了解以进行设计和部署。 在本节中，我们将介绍DL和相关的技术术语，为讨论DL和边缘计算的集成铺平道路（更多详细信息，请参见[66]）。

A. Neural Networks in Deep Learning

DL模型由各种类型的深度神经网络（DNN）组成[66]。 DNA的基本结构和功能的基本原理介绍如下。

1) Fully Connected Neural Network (FCNN)：如图7（a）所示，FCNN每一层的输出，即多层感知器（MLP），被前馈到下一层。在相邻的FCNN层之间，神经元（单元）的输出（输入单元或隐藏单元）直接传递到属于下一层的神经元并由其激活[67]。 FCNN可以用于特征提取和函数逼近，但是具有很高的复杂性，适度的性能和缓慢的收敛性。

2) Auto-Encoder (AE): AE，如图7（b）所示，实际上是两个NN的堆栈，它们以一种不受监督的学习风格将输入复制到其输出。 第一个NN学习输入（编码）的代表特征。 第二个NN将这些特征用作输入，并在匹配输入输出像元处恢复原始输入的近似值，用于从输入到输出收敛于身份函数，作为最终输出（解码）。 由于AE能够学习输入数据的低维有用特征以恢复输入数据，因此它通常用于分类和存储高维数据[68]。

3) Convolutional Neural Network (CNN): 通过使用合并操作和一组不同的移动滤波器，CNN会抓住相邻数据块之间的相关性，然后生成输入数据的连续更高级别的抽象，如图7（c）所示。 与FCNN相比，CNN可以在提取特征的同时降低模型的复杂性，从而降低了过拟合的风险[69]。 这些特性使CNN在图像处理中获得了卓越的性能，并且在处理类似于图像的结构数据时也很有用。

4) Generative Adversarial Network (GAN): GAN起源于博弈论。 如图7（d）所示，GAN由生成器和鉴别器组成。 生成器的目标是通过有意识地在反馈输入单元中引入反馈，从而尽可能地了解真实数据的分布，而鉴别器则是正确确定输入数据是来自真实数据还是来自生成器。 这两个参与者需要不断优化其在对抗过程中产生和区分的能力，直到找到纳什均衡[70]为止。 根据从真实信息中学到的特征，训练有素的生成器可以因此制造出难以区分的信息。

5) Recurrent Neural Network (RNN): RNN设计用于处理顺序数据。 如图7（e）所示，RNN中的每个神经元不仅从上层接收信息，而且从其自己的前一个通道接收信息[10]。 通常，RNN是预测未来信息或恢复顺序数据缺失部分的自然选择。 但是，RNN的一个严重问题是梯度爆炸。 如图7（f）所示，LSTM通过添加门结构和定义良好的存储单元来改善RNN，可以通过控制（禁止或允许）信息流来克服此问题[71]。

B. Deep Reinforcement Learning (DRL)

如图8所示，RL的目标是使环境中的代理能够在当前状态下采取最佳行动，以最大化长期收益，其中，代理的行动与通过环境的状态之间的交互被建模为马尔可夫决策过程（MDP）。 DRL是DL和RL的结合，但它更侧重于RL，旨在解决决策问题。 DL的作用是利用DNN的强大表示能力来拟合值函数，或者使用直接策略来解决状态动作空间或连续状态动作空间爆炸的问题。 凭借这些特性，DRL成为机器人技术，财务，推荐系统，无线通信等方面的强大解决方案[18]，[72]。

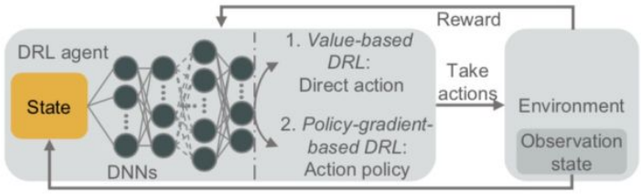

Fig. 8. Value-based and policy-gradient-based DRL approaches.

1) Value-based DRL: 作为基于值的DRL的代表，深度Q学习（DQL）使用DNN来拟合动作值，成功地将高维输入数据映射到动作[73]。 为了确保训练的稳定收敛，采用经验重播的方法打破过渡信息之间的相关性，并建立了一个单独的目标网络来抑制不稳定。 此外，双重深度Q学习（Double-DQL）可以处理DQL通常高估动作值的问题[74]，而决斗深度Q学习（Dueling-DQL）[75]可以了解哪些状态有价值（或不有价值） 无需了解每个动作在每个状态下的效果。

2) Policy-gradient-based DRL: 策略梯度是另一种常见的策略优化方法，例如深度确定性策略梯度（DDPG）[76]，异步先驱关键算法（A3C）[77]，近策略优化（PPO）[78]等。 它通过连续计算策略期望奖励相对于它们的梯度来更新策略参数，最后收敛到最优策略[79]。 因此，在解决DRL问题时，可以使用DNN对策略进行参数化，然后通过策略梯度法对其进行优化。 此外，基于策略梯度的DRL中广泛采用了Actor-Critic（AC）框架，其中策略DNN用于更新策略，与Actor相对应。 值DNN用于近似状态动作对的值函数，并提供与Critic对应的梯度信息。

C. Distributed DL Training

目前，集中训练DL模型消耗大量时间和计算资源，阻碍了算法性能的进一步提高。 但是，分布式培训可以通过充分利用并行服务器来促进培训过程。 有两种执行分布式训练的常用方法，即数据并行性和模型并行性[80]-[83]，如图9所示。

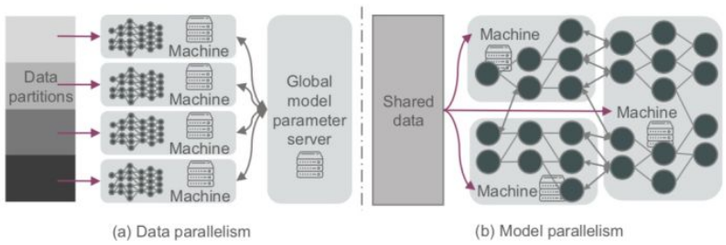

Fig. 9. Distributed training in terms of data parallelism and model parallelism.

模型并行性首先将大型DL模型拆分为多个部分，然后提供数据样本以并行训练这些分段模型。 这不仅可以提高训练速度，而且可以处理模型大于设备内存的情况。 训练大型DL模型通常需要大量的计算资源，甚至需要数千个CPU来训练大规模DL模型。 为了解决这个问题，可以将分布式GPU用于模型并行训练[84]。 数据并行性是指将数据划分为多个分区，然后分别与自己分配的数据样本并行地训练模型副本。 通过这种方式，可以提高模型训练的训练效率[85]。

巧合的是，分散了许多终端设备，边缘节点和云数据中心，并设想通过边缘计算网络进行连接。 一旦DL培训跳出云端，这些分布式设备就可能成为强大的贡献者。

D. Transfer Learning (TL)

TL可以将知识从源域转移到目标域，如图7（f）所示，以便在目标域中获得更好的学习性能[86]。 通过使用TL，可以将大量计算资源学到的现有知识转移到新场景中，从而加快了训练过程并降低了模型开发成本。 最近，出现了一种新型的TL，即知识蒸馏（KD）[87]。 如图7（h）所示，KD可以从训练有素的模型（教师）中提取隐式知识，其推理具有出色的性能，但需要较高的开销。 然后，通过设计目标DL模型的结构和目标函数，将知识“转移”到较小的DL模型（学生），以便显着减少（修剪或量化）的目标DL模型获得尽可能高的性能。

E. Potential DL Frameworks for the Edge

DL模型的开发和部署依赖于各种DL框架的支持。 但是，不同的DL框架具有各自的应用场景。 为了在边缘上以及边缘部署DL，需要高效的轻量级DL框架。 表III中列出了可能支持未来边缘智能的DL框架的功能（不包括如Theano [88]等对边缘设备不方便或不可

用的框架）。

TABLE III
POTENTIAL DL FRAMEWORKS FOR EDGE COMPUTING

Framework	Owner	Support Edge	Android	iOS	Arm-linux	FPGA	DSP	GPU	Mobile GPU	Support Training
CNTK [53]	Microsoft	×	×	×	×	×	×	✓	×	✓
Chainer [54]	Preferred Networks	×	×	×	×	×	×	✓	×	✓
TensorFlow [55]	Google	✓	×	×	✓	×	×	✓	×	✓
DL4J [56]	Skymind	✓	✓	×	✓	×	×	✓	×	✓
TensorFlow Lite [57]	Google	✓	✓	×	✓	×	×	✓	✓	×
MXNet [58]	Apache Incubator	✓	✓	✓	✓	×	×	✓	×	✓
(Py)Torch [59]	Facebook	✓	✓	✓	✓	✓	×	✓	×	✓
CoreML [60]	Apple	✓	×	✓	×	×	×	×	✓	×
SNPE [50]	Qualcomm	✓	✓	×	✓	×	✓	×	✓	×
NCNN [61]	Tencent	✓	✓	✓	✓	×	×	×	✓	×
MNN [62]	Alibaba	✓	✓	✓	✓	×	×	×	✓	×
Paddle-Mobile [63]	Baidu	✓	✓	✓	✓	✓	×	×	×	×
MACE [64]	XiaoMi	✓	✓	✓	✓	×	×	×	✓	×

IV. DEEP LEARNING APPLICATIONS ON EDGE

通常，由于大多数DL模型复杂且难以在资源受限的设备方面计算其推理结果，因此DL服务当前部署在云数据中心（云）中以处理请求。 但是，这种“端云”架构无法满足实时DL服务的需求，例如实时分析，智能制造等。因此，在边缘部署DL应用程序可以扩大DL的应用场景，尤其是在 关于低延迟特性。 在下文中，我们将介绍与DL和边缘计算相关的应用程序以及它们的元素如何系统地组织，突出了它们比没有边缘计算的体系结构的优势。

A. Real-time Video Analytic

实时视频分析在各个领域都很重要，例如自动驾驶，VR和增强现实（AR），智能监控等。总的来说，为其应用DL需要大量的计算和存储资源。 不幸的是，在云中执行这些任务通常会导致高带宽消耗，意外的延迟和可靠性问题。 随着边缘计算的发展，这些问题倾向于通过将视频分析移至靠近数据源（即终端设备或边缘节点）作为云的补充物来解决。 在本节中，如图10所示，我们将相关工作总结为混合层次结构，分为三个层次：末端，边缘和云。

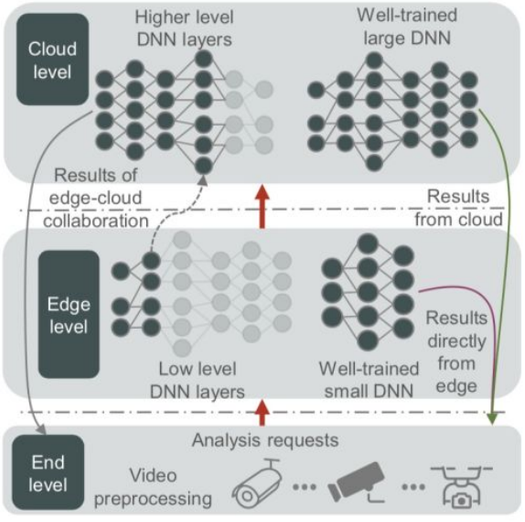

Fig. 10. The collaboration of the end, edge and cloud layer for performing real-time video analytic by deep learning.

1) End Level: 在end level，诸如智能手机和监控摄像机之类的视频捕获设备负责视频捕获，媒体数据压缩[89]，图像预处理和图像分割[90]。 通过与这些参与设备的协调，与领域受限的深度模型一起使用时，协同训练一个领域感知的适应模型可以提高对象识别的准确性[91]。 此外，为了将DL计算适当地分流到终端设备，边缘节点或云，终端设备应全面考虑视频压缩和关键指标（例如网络条件，数据使用，电池消耗，处理延迟）之间的权衡。 帧速率和分析的准确性，从而确定最佳的卸载策略[89]。

*2）Edge Level*边缘级别的许多分布式边缘节点通常相互协作以提供更好的服务。 例如，LAVEA [92]将边缘节点连接到同一接入点或BS以及终端设备，这确保了服务可以像Internet访问一样普遍存在。另外，由于云更适合DL训练，而通过在边缘执行逻辑推理可以使DL推理受益，因此EdgeEye [93]分离DL模型，并将这些分区分配给不同级别的终端设备和边缘节点。除了边缘协作之外，在边缘上压缩DL模型还可以提高整体性能。 如[94]中所述，通过减少CNN层中不必要的过滤器，可以在确保分析性能的同时大大减少边缘层的资源消耗。

*3) Cloud Level*在cloud level，云负责边缘层之间的DL模型的集成并更新边缘节点上分布式DL模型的参数[89]。 由于边缘节点的本地知识可能会大大削弱边缘节点上的分布式模型训练性能，因此云需要集成不同的训练有素的DL模型以获取全局知识。 当边缘无法可靠地提供服务时（例如，以低置信度检测对象），云可以使用其强大的计算能力和全局知识进行进一步处理，并协助边缘节点更新DL模型。

B. Autonomous Internet of Vehicles (IoVs)

可以预见，可以连接车辆以提高安全性，提高效率，减少事故并减少交通运输系统中的交通拥堵[95]。 尽管通常分别进行研究，但是有许多信息和通信技术，例如联网，缓存，边缘计算，可用于促进IoV。 一方面，边缘计算为车辆提供了低延迟，高速通信和快速响应服务，从而使自动驾驶成为可能。 另一方面，DL技术在各种智能车辆应用中很重要。 此外，他们有望优化复杂的IoV系统。

在[95]中，提出了一个整合这些技术的框架。 这种集成的框架能够动态分配网络，缓存和计算资源，以满足不同车辆应用的需求[95]。 由于该系统涉及多维控制，因此首先使用基于DRL的方法来解决优化问题，以增强整体系统性能。 类似地，在[96]中也使用DRL来获得车辆边缘计算中的最佳任务卸载策略。 此外，可以利用V2V（车对车）通信技术进一步连接车辆，作为边缘节点或基于DRL的控制策略管理的终端设备[97]。

C. Intelligent Manufacturing

智能制造时代的两个最重要的原理是自动化和数据分析，前一个是主要目标，后一个是最有用的工具之一[98]。 为了遵循这些原则，智能制造应首先解决响应延迟，风险控制和隐私保护，因此需要DL和边缘计算。 在智能工厂中，边缘计算有利于将计算资源，网络带宽和云的存储容量扩展到IoT边缘，并在制造和生产过程中实现资源调度和数据处理[99]。 对于自主制造检查，aDeepIns [98]使用DL和边缘计算分别保证性能和过程延迟。 该系统的主要思想是划分用于检查的DL模型，并将其分别部署在端，边缘和云层上，以提高检查效率。

然而，随着物联网边缘设备的指数增长，1）如何远程管理不断发展的DL模型，以及2）如何为它们持续评估这些模型是必要的。在[100]中，开发了一个框架来应对这些挑战， 支持智能制造期间的复杂事件学习，从而促进物联网边缘设备上实时应用程序的开发。 此外，还应考虑物联网边缘设备[101]的功率，能效，内存占用限制。 因此，可以集成缓存，与异构物联网设备的通信以及计算分流[102]，以打破资源瓶颈。

D. Smart Home and City

物联网的普及将为家庭生活带来越来越多的智能应用，例如智能照明控制系统，智能电视和智能空调。 但是，与此同时，智能家居需要在角落，地板和墙壁上部署大量无线物联网传感器和控制器。 为了保护敏感的家庭数据，智能家庭系统的数据处理必须依靠边缘计算。 像[103]，[104]中的用例一样，部署了边缘计算来优化室内定位系统和家庭入侵监控，以便与使用云计算相比，它们可以获得更低的延迟，并且具有更高的准确性。 此外，DL和边缘计算的结合可以使这些智能服务变得更加多样化和强大。 例如，它赋予机器人动态视觉服务的能力[105]并启用有效的音乐认知系统[106]。

如果将智能家居扩展到社区或城市，则公共安全，健康数据，公共设施，交通和其他领域将受益。在智能城市中应用边缘计算的最初意图更多是出于成本和效率方面的考虑。城市中地理分布的数据源的自然特性要求基于边缘计算的范例来提供位置感知和对时延敏感的监视和智能控制。例如，[107]中的分层分布式边缘计算架构可以支持未来智能城市中大规模基础设施组件和服务的集成。这种体系结构不仅可以支持终端设备上对延迟敏感的应用程序，而且可以在边缘节点上有效地执行稍有延迟的任务，而负责深度分析的大规模DL模型则托管在云中。此外，DL可用于协调和调度基础结构，以在城市的某个区域（例如，校园内[108]）或整个城市之间实现整体负载平衡和最佳资源利用。

V. DEEP LEARNING INFERENCE IN EDGE

为了进一步提高准确性，DNN越来越深，需要更大规模的数据集。通过这种方法，将引入巨大的计算成本。当然，DL模型的出色性能离不开高级硬件的支持，并且难以在资源有限的边缘部署它们。因此，大型DL模型通常部署在云中，而终端设备只是将输入数据发送到云中，然后等待DL推断结果。但是，仅基于云的推理限制了DL服务的普遍部署。具体而言，它不能保证实时服务的延迟要求，例如具有严格等待时间要求的实时检测。此外，对于重要的数据源，应解决数据安全和隐私保护问题。为了解决这些问题，DL服务倾向于求助于边缘计算。因此，应进一步定制DL模型以适合资源受限的边缘，同时仔细处理推理精度和执行延迟之间的权衡。

A. Optimization of DL Models in Edge

DL任务通常需要大量计算，并且需要占用大量内存。 但是在边缘，没有足够的资源来支持原始的大规模DL模型。 优化DL模型并量化其权重可以降低资源成本。 实际上，模型冗余在DNN中很常见[109]，[110]，并且可以用来使模型优化成为可能。最重要的挑

战是如何确保优化后不会造成模型准确性的重大损失。 换句话说，优化方法应该转换或重新设计DL模型，并使它们适合边缘设备，并尽可能减少模型性能损失。 在本节中，将讨论针对不同场景的优化方法：1）具有相对足够资源的边缘节点的常规优化方法； 2）资源预算紧张的终端设备的细粒度优化方法。

*1) General Methods for Model Optimization:*一方面，以几乎恒定的计算开销增加DL模型的深度和宽度是优化的一个方向，例如CNN的初始[111]和深度残差网络[112]。 另一方面，对于更通用的神经网络结构，现有的优化方法可以分为四类[113]：1）参数修剪和共享[114]，[115]，还包括权重量化[116]-[118]；2）低阶分解[109]； 3）转移/紧凑的卷积过滤器[94]，[119]，[120]； 4）知识蒸馏[121]。 这些方法可以应用于不同种类的DNN，也可以组合起来优化边缘的复杂DL模型。

*2) Model Optimization for Edge Devices.*除了有限的计算和内存占用空间外，还需要考虑其他因素，例如网络带宽和功耗。 在本节中，将区分和讨论在边缘设备上运行DL的工作。

• *Model Inputs.* 每个应用程序场景都有特定的优化空间。 例如，对于实时人类检测，通过缩小分类器的搜索空间（如图11所示）以专注于监视视频帧中的人类对象，CNN模型因此可以在边缘设备上运行以 检测行人[122]。 这种方法在不改变神经网络结构的情况下极大地压缩了模型输入的大小，可以大大减少所需的计算开销。 但是同时，还需要对相关的应用场景有深入的了解，才能挖掘出潜在的优化空间。

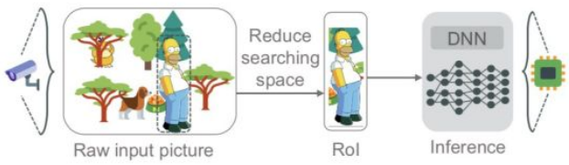

Fig. 11. Optimization for model inputs, e.g., narrowing down the searching space of DL models (pictures are with permission from [123]).

• *Model Structures:*不关注特定的应用程序，而是关注广泛使用的DNN的结构也是可行的。 通过这种方式，逐点分组卷积和信道混洗[124]，并行卷积和池计算[125]，深度可分离卷积[94]可以在保持精度的同时大大降低计算成本。 此外，如图12所示，参数修剪也可以自适应地应用于模型结构优化[126]-[128]。 此外，如果跨越算法，软件和硬件之间的边界，优化可能会更加有效。 具体而言，通用硬件尚未为模型优化引入的不规则计算模式做好准备。 因此，如[126]中所述，硬件体系结构应设计为可直接用于优化模型。

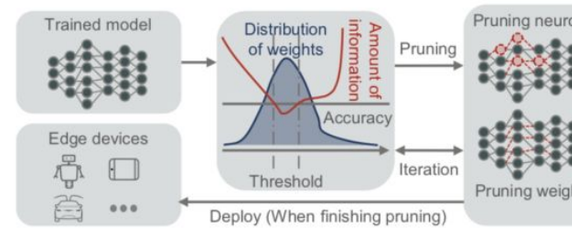

Fig. 12. Adaptive parameters pruning in model structure optimization.

• *Model Frameworks:*考虑到DL的高内存占用量和计算需求，在边缘设备上运行它们需要专家量身定制的软件和硬件框架。 如果一个软件框架是有价值的，则它是：1）提供优化的软件内核库以支持DL的部署[129]； 2）通过找到非冗余隐藏元素的最小数量，将DL模型自动压缩为较小的密集矩阵[130]； 3）对所有常用的DL结构[127]，[130]，[131]进行量化和编码； 4）专门针对上下文的DL模型，并在多个同时执行的DL模型之间共享资源[131]。 在硬件方面，与DRAM（动态RAM）相比，在SRAM（静态随机存取存储器）上运行DL模型可实现更好的节能效果[127]。 因此，如果底层硬件直接支持在片上SRAM上运行优化的DL模型[132]，则DL性能将受益。

B. Segmentation of DL Models in Edge

在[12]中，在云和边缘设备上评估了最先进的DL模型的延迟和功耗，发现将数据上传到云是当前DL服务方法的瓶颈（导致传输开销很大）。 划分DL模型并执行分布式计算可以实现更好的端到端延迟性能和能效。此外，通过将部分DL任务从云中推到边缘，可以提高云的吞吐量。 因此，可以将DL模型划分为多个分区，然后分配给1）终端设备[133]上的异构本地处理器（例如GPU，CPU），2）分布式边缘节点[134]，[135]或3）协作式“端到端云”架构[12]，[46]，[136]，[137]。

最常见的分割方法是对DL模型进行水平分割，即沿末端，边缘和云进行分割。 挑战在于如何智能地选择分区点。 如图13所示，确定分区点的一般过程可以分为三个步骤[12]，[136]：1）测量和建模不同DNN层的资源成本以及之间的中间数据大小 层; 2）通过特定的层配置和网络带宽预测总成本； 3）根据延迟，能量需求等从候选分区中选择最佳方案。另一种模型分割是垂直分区，尤其是对于CNN [135]。 对于水平分区，垂直分区将各层融合在一起，并以网格方式在垂直方向上对其进行分区，从而将CNN层划分为独立可分配的计算任务。

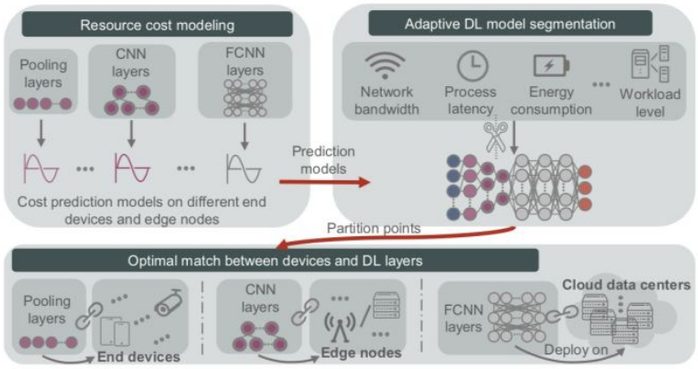

Fig. 13. Segmentation of DL models in the edge.

C. Early Exit of Inference (EEoI)

为了在模型精度和处理延迟之间达到最佳平衡，可以为每个DL服务维护具有不同模型性能和资源成本的多个DL模型。 然后，通过智能选择最佳模型，可以实现所需的自适应推断[138]。 但是，这种想法可以通过出现的EEoI进一步改进[139]。

DNN中附加层的性能提升是以增加前馈推理的等待时间和能耗为代价的，随着DNN越来越大和越来越深入，这些成本对于边缘设备运行实时且对能量敏感的DL应用程序变得更加困难。 对于部分样本，附加的侧分支分类器，如果具有较高的置信度，则EEoI允许推断通过这些分支提前退出。 对于更困难的样本，EEoI将使用更多或所有DNN层来提供最佳预测。

如图14所示，通过利用EEoI，可以在边缘设备上使用DL模型的浅层部分进行快速而局部的推断。通过这种方式，边缘设备上的浅层模型可以快速执行初始特征提取，并且如果有把握，可以直接给出推断结果。否则，部署在云中的其他大型DL模型将执行进一步的处理和最终推断。与直接将DL计算卸载到云相比，这种方法比边缘设备上修剪或量化的DL模型具有更低的通信成本并且可以实现更高的推理精度[98]，[140]。此外，由于仅将即时功能而非原始数据发送到云，因此它提供了更好的隐私保护。然而，不应将EEoI视为独立于模型优化（第V-A2节）和分段（第V-B节）。在端部，边缘和云上分布式DL的构想应考虑到它们的协作，例如，开发用于自适应DNN分区和EEoI的协作式和按需协同推理框架[141]。

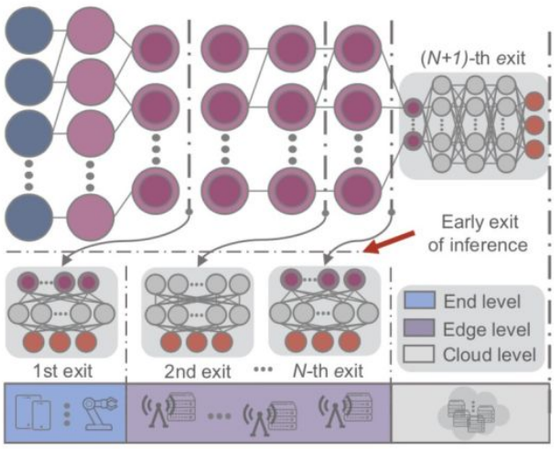

Fig. 14. Early exit of inference for DL inference in the edge.

VI. EDGE COMPUTING FOR DEEP LEARNING

DL服务（尤其是移动DL）的广泛部署需要边缘计算的支持。 这种支持不仅在网络体系结构级别，边缘硬件和软件的设计，适配和优化同样重要。 具体来说，1）定制的边缘硬件以及相应的优化软件框架和库可以帮助更有效地执行DL； 2）边缘计算架构可以实现DL计算的卸载。 3）设计良好的边缘计算系统可以更好地维护在边缘运行的DL服务； 4）用于评估边缘DL性能的公平平台有助于进一步实现上述实现。

A. Edge Hardware and Software Stacks for DL

*1) Mobile CPUs and GPUs:*如果直接在事件发生位置附近的轻型边缘设备（例如手机，可穿戴设备和监控摄像头）上启用DL应用程序，则它们的价值将更高。 作为边缘设备一部分的低功耗IoT设备可用于进行轻量级DL计算，从而避免与云进行通信，但仍需要面对有限的计算资源，内存占用空间和能耗。 为了缓解这些瓶颈，在[125]中，对ARM Cortex-M微控制器进行了研究，以使其成为DL潜在可行的边缘硬件。 通过开发的高效NN内核CMSIS-NN集合，可以最小化NN在ARM Cortex-M处理器内核上的内存占用，然后将DL模型安装到IoT设备中，同时实现正常的性能和能效。

DeepMon [142]作为在移动GPU上处理CNN层的一套优化（在硬件和软件上），可以通过利用第一人称视角中连续帧之间的相似性来大大减少推理时间。 视频。 这样的特征在CNN层中引起大量的计算重复，从而激发了缓存CNN层的计算结果的想法。 此外，借助矩阵分解，CNN层中的高维矩阵运算（尤其是乘法）在移动GPU中变得可用并可以加速。 鉴于这项工作，可以使用特定的DL模型潜在地探索已经部署在边缘设备中的各种移动GPU，它们在启用边缘智能中起着更重要的作用。

除了推论[125]，[142]以外，在[143]中还讨论了影响移动CPU和GPU上的DL训练性能的重要因素。 由于常用的DNN模型（例如VGG [144]）对于主流边缘设备的内存大小而言太大，因此采用相对较小的Mentee网络[145]来评估DL训练。 评估结果指出，DL模型的大小对于训练性能至关重要，而移动CPU和GPU的有效融合对于加速训练过程至关重要。

*2) FPGA-based Solutions:*尽管GPU解决方案已在云中广泛用于DL训练和推理，但是受边缘强大的功能和成本预算限制，这些解决方案可能不可用。 此外，边缘节点应该能够一次满足多个DL计算请求，这使得仅使用轻量级CPU和GPU不切实际。 因此，探索了基于FPGA（现场可编程门阵列）的边缘硬件，以研究其用于边缘DL的可行性。

基于FPGA的边缘设备可以通过任意大小的卷积和可重新配置的池来实现CNN加速[125]，并且相对于基于RNN的语音识别应用程序，它们的性能比最新的CPU和GPU实现[126]快。 实现更高的能源效率。 在[49]中，开发了基于FPGA的边缘计算平台的设计和设置，以支持从移动设备到边缘FPGA平台的DL计算卸载。 在实现基于FPGA的边缘时，将无线路由器和FPGA板组合在一起。 在典型的视觉应用中测试该初步系统后，基于FPGA的边缘平台在能耗和硬件成本方面均比基于GPU（或CPU）的边缘更具优势。

尽管如此，FPGA和GPU / CPU还是更多如表IV所示，适用于边缘计算的尚待决仍在待决。 在[146]中进行了详尽的实验，以研究采用FPGA进行GPU边缘计算的优势：1）能够提供对工作负载不敏感的吞吐量； 2）保证高并发DL计算的一致高性能； 3）更好的能源利用效率。 但是，FPGA的缺点在于，大多数程序员都不熟悉在FPGA上开发高效的DL算法。 尽管诸如Xilinx SDSoC之类的工具可以大大降低难度[49]，但至少到目前为止，仍需要进行其他工作才能将针对GPU编程的最新DL模型移植到FPGA平台中。

*3) Performance Evaluation for Edge DL:*在选择适当的边缘硬件和相关的软件堆栈以在边缘上部署各种DL服务的整个过程中，有必要评估它们的性能。此外，英制评估方法可以指出可能的方向，以针对特定边缘硬件优化软件堆栈。

在[147]中，当在资源受限的边缘设备上执行DL推理时，DL库的性能首次得到了评估，该性能与诸如延迟，内存占用量和能源之类的指标有关。另外，特别是对于Android智能手机，作为一种带有移动CPU或GPU的边缘设备，AI Benchmark [51]广泛研究了各种设备配置上的DL计算能力。实验结果表明，没有一个DL库或硬件平台可以完全胜过其他库，而加载DL模型可能比执行它花费更多的时间。这些发现表明，仍有机会进一步优化边缘硬件，边缘软件堆栈和DL库的融合。

TABLE IV
COMPARISON OF SOLUTIONS FOR EDGE NODES

Metrics	Preferred Hardware	Analysis
Resource overhead	FPGA	FPGA can be optimized by customized designs.
DL training	GPU	Floating point capabilities are better on GPU.
DL inference	FPGA	FPGA can be customized for the specific DL model.
Interface scalability	FPGA	It is more free to implement interfaces on FPGAs.
Space occupation	CPU/FPGA	Lower power consumption of FPGA leads to smaller space occupation.
Compatibility	CPU/GPU	CPUs and GPUs have more stable architecture.
Development efforts	CPU/GPU	Toolchains and software libraries facilitate the practical development.
Energy efficiency	FPGA	Customized designs can be optimized.
Concurrency support	FPGA	FPGAs are suitable for stream processing.
Timing latency	FPGA	Timing on FPGAs can be an order of magnitude faster than GPUs.

B. Edge Computing Mode for DL Computation

DL带来了终端设备负担不起的密集计算任务。 边缘计算的概念可以通过将DL计算从终端设备转移到边缘节点来潜在地解决这一难题。 伴随着边缘架构，以DL为中心的边缘节点可以成为云计算基础架构的重要扩展，以处理大量DL任务。 在本节中，我们对用于DL计算的四种边缘计算模式进行了分类，如图15所示。

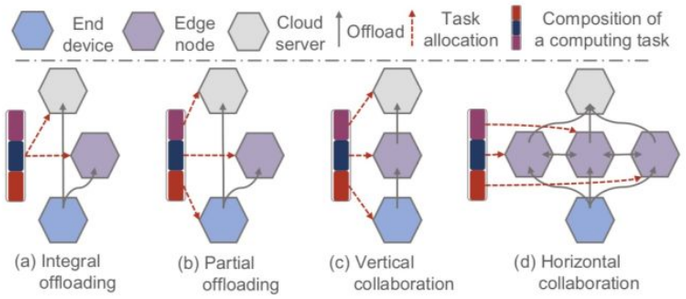

Fig. 15. Edge computing modes for DL computation.

1) Integral Offloading: DL计算卸载的最自然模式类似于现有的“端云”计算，即，终端设备将其计算请求发送到云以获取DL推理结果（如图15（a）所示）。通过从DL任务分解和资源优化的组合问题中解脱出来（可以带来额外的计算成本和调度延迟），这种卸载很简单，因此易于实现。

在[148]中，提出了一种分布式基础架构，它将强大的边缘节点与功能较弱的终端设备联系在一起。 DL推理可以在末端或边缘执行，具体取决于所需DL准确性，推理延迟，电池电量和网络条件之间的权衡。对于每个DL任务，终端设备将决定是本地处理还是将其完全卸载到边缘节点。

此外，在卸载问题中不应忽略边缘节点之间的工作负载优化，因为与云相比，边缘节点通常受到资源限制。为了满足在边缘资源有限的情况下完成DL任务的延迟和能量需求，可以采用在边缘提供不同模型大小和性能的DL模型来完成一种任务。因此，可以在边缘节点上部署可分别处理不同DL模型的多个虚拟机（VM）或容器，以处理DL请求。具体而言，当具有较低复杂度的DL模型可以满足要求时，将选择它作为服务模型。通过优化VM的工作量分配权重和计算能力，可以开发旨在降低能源成本和延迟的优化模型[149]，同时保证DL推理的准确性。

2) Partial Offloading: 将DL任务部分卸载到边缘也是可行的（如图15（b）所示）。可以开发一个卸载系统，以实现DL任务的在线细粒度划分，并确定如何将这些划分的任务分配给终端设备和边缘节点。如[150]所示，所提出的能够自适应划分通用计算机程序

的MAUI通过在网络约束下优化任务分配策略，可以节省一个数量级的能量。更重要的是，该解决方案可以在运行时分解整个程序，而无需在部署程序之前手动划分程序员。尽管这项工作没有考虑DL应用程序，但仍然可以部分了解DL任务的卸载潜力。

在任何边缘节点上预装DL模型以处理来自终端设备的各种请求（特别是在考虑到移动性时）是不现实的，并且将所有DL模型填充到每个边缘节点中甚至是不切实际的。因此，当未预先安装DL模型时，终端设备应首先将其DL模型上传到边缘节点。不幸的是，由于上传时间长，它可能严重延迟卸载的DL计算。

为了应对这一挑战，在[151]中提出了一种增量卸载系统IONN。与打包整个DL模型以进行上载不同，IONN将准备上载的DL模型分为多个分区，然后按顺序将其上载到边缘节点。另一方面，接收分区模型的边缘节点在每个分区模型到达时逐步构建DL模型，同时即使在上载整个DL模型之前也能够执行卸载的部分DL计算。因此，关键在于确定最佳DL分区和上载顺序。具体地，一方面，优先选择性能收益高，上传开销低的DNN层，从而使边缘节点快速构建部分DNN，以达到最佳的查询性能。另一方面，不会带来任何性能提升的不必要的DNN层将不会上载，从而避免了卸载。

3) Vertical Collaboration: 如第VI-B1和VI-B2节所讨论的，在“端点”架构中预期的卸载策略对于支持较少计算密集型DL服务和小规模并发DL查询是可行的。但是，当需要一次处理大量DL查询时，单个边缘节点肯定不足。

协作的自然选择是，当DL任务卸载时，边缘执行数据预处理和初步学习。然后，中间数据，即边缘体系结构的输出，被传输到云中以进行进一步的DL计算[152]。但是，可以进一步挖掘DNN的层次结构，以适应垂直协作。在[12]中，根据数据和计算特性，在终端设备和边缘节点上对DNN的所有层进行了配置，以生成性能预测模型。基于这些预测模型，无线条件和服务器负载水平，拟议的Neurosurgeon根据端到端延迟或移动能耗评估每个候选点，并将DNN划分为最佳。然后，它决定DNN分区的分配，即应在终端，边缘或云上部署哪一部分，同时实现终端设备的最佳延迟和能耗。

通过利用EEoI（V-C节），可以更有效地适应垂直协作。 DNN的分区可以映射到分布式计算层次结构（即末端，边缘和云），并且可以使用多个早期出口点进行训练[140]。因此，末端和边缘可以对自己执行一部分DL推断，而不是直接请求云。使用推断后的出口点，可以给出本地设备有信心的DL任务的结果，而无需将任何信息发送到云。为了提供更准确的DL推断，中间DNN输出将通过使用其他DNN层发送到云中以进行进一步推断。尽管如此，中间输出（例如高分辨率监视视频流）应经过精心设计，比原始输入小得多，因此可以大大减少端与边缘（或边缘与云）之间所需的网络流量。

尽管垂直协作可以看作是云计算的发展，但也就是“端云”策略。 与纯粹的“端到端”策略相比，垂直协作的过程可能会延迟，因为它需要与云进行额外的通信。 但是，纵向协作有其自身的优势。 一方面，当边缘架构无法独自承担大量DL查询时，云架构可以共享部分计算任务，从而确保为这些查询提供服务。 另一方面，原始数据必须先在边缘进行预处理，然后再传输到云中。 如果这些操作可以大大减少中间数据的大小，从而减少网络流量，则可以减轻骨干网的压力。

*4) Horizontal Collaboration*在第VI-B3节中，讨论了纵向协作。但是，边缘或端部之间的设备也可以在没有云的情况下进行组合，以处理需要大量资源的DL应用程序，即水平协作。通过这种方式，可以对训练后的DNN模型或整个DL任务进行分区，并将其分配给多个终端设备或边缘节点，以通过减轻每个终端设备或边缘节点的资源成本来加速DL计算。在[153]中提出的MoDNN通过无线局域网（WLAN）在本地分布式移动计算系统中执行DL。 DNN的每一层都被划分为多个切片以提高并行度并减少内存占用，并且这些切片逐层执行。通过多个终端设备之间的执行并行性，可以显着加速DL计算。

对于特定的DNN结构（例如CNN），可以应用更精细的网格分区以最小化通信，同步和内存开销[115]。在[135]中，提出了一种能够将每个CNN层划分为独立可分配任务的融合切片分区（FTP）方法。与[12]中仅按层划分DNN相比，FTP可以融合层并以网格方式垂直对DNN进行分区，因此，无论分区和设备的数量如何，都可以将参与的边缘设备所需的内存占用空间降至最低，同时减少通信量以及任务迁移成本。此外，为了支持FTP，分布式工作窃取运行时系统，即空闲边缘设备从具有活动工作项的其他设备窃取任务[135]，可以自适应地分配FTP分区以平衡协作边缘设备的工作量。

VII. DEEP LEARNING TRAINING AT EDGE

当前在云数据中心的DL培训（无论是否分发），即云培训或云边缘培训[47]，即在边缘处对培训数据进行预处理，然后传输到云，并不适合所有类型的DL服务，特别是对于需要局部性和持续训练的DL模型。此外，如果需要将大量数据从分布式终端设备或边缘节点连续传输到云，则将消耗大量的通信资源，从而使无线和骨干网恶化。例如，对于集成了目标检测和目标跟踪的监控应用，如果终端设备直接将大量实时监控数据发送到云中进行持续培训，则将带来很高的网络成本。此外，将所有数据合并到云中可能会违反隐私问题。所有这些挑战提出了针对现有云培训的新颖培训方案的需求。

自然地，由大量具有适度计算资源的边缘节点组成的边缘体系结构可以通过处理数据或自行训练来缓解网络的压力。将边缘作为培训的核心架构，在边缘或“边缘云”中进行培训称为“ DL at Edge”。这种DL训练可能需要大量资源来消化分布式数据并交换更新。

A. Distributed Training at Edge

边缘的分布式训练可以追溯到[154]的工作，其中提出了一种用于边缘计算网络的分散SGD（随机梯度下降）方法来解决大型线性回归问题。然而，由于用于训练大规模DL模型的通信成本非常高，因此该方法是为地震成像应用而设计的，无法推广到未来的DL训练中。在[155]中，提出了两种针对边缘计算环境的分布式学习解决方案。如图16所示，一种解决方案是每个终端设备都基于本地数据训练模型，然后在边缘节点上聚合这些模型更新。另一个是边缘节点训练自己的局部模型，并交换和更新其模型更新以构建全局模型。尽管在边缘进行大规模的分布式训练可以避免将庞大的原始数据集传输到云中，但不可避免地会引入边缘设备之间进行梯度交换的通信成本。此外，在实践中，边缘设备可能会遭受更高的等待时间，更低的传输速率和间歇性连接，因此进一步阻碍了属于不同边缘设备的DL模型之间的梯度交换。

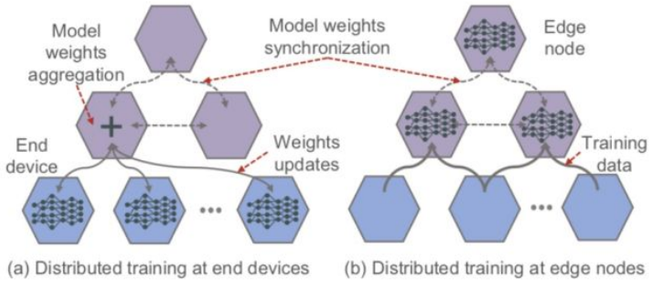

Fig. 16. Distributed DL training at edge environments.

大多数梯度交换是多余的，因此可以压缩更新的梯度以降低通信成本，同时保持训练的准确性（例如[156]中的DGC）。首先，DGC规定仅交换重要的梯度，即，仅传输大于启发式给定阈值的梯度。为了避免信息丢失，其余的梯度将在本地累积，直到超过阈值为止。要注意的是，将对梯度进行立即编码或压缩，以便立即进行传输或累积以进行后续交换，从而节省了通信成本。其次，考虑到稀疏的梯度更新可能会损害DL训练的收敛性，因此采用了动量校正和局部梯度修剪来减轻潜在风险。通过动量校正，稀疏更新可以近似等于密集更新。在将当前梯度添加到本地每个边缘设备上的先前累积之前，执行梯度修剪以避免梯度累积可能引入的爆炸梯度问题。当然，由于部分渐变会延迟更新，因此可能会降低收敛速度。因此，最后，为了防止过时的动量危害训练的性能，应停止延迟梯度的动量，并在训练开始时采用较小的主动学习率和梯度稀疏性，以减少延迟的极端梯度的数量。

为了减少在分布式训练期间同步梯度和参数的通信成本，可以将两种机制组合在一起[157]。第一种是通过利用稀疏的训练梯度来传输仅重要的梯度[158]。保持隐藏权重以记录参与梯度同步的梯度坐标的时间，并且具有较大隐藏权重值的梯度坐标被视为重要的梯度，并且很可能在下一轮训练中被选择。另一方面，如果直接忽略残余梯度坐标（即次要梯度），将严重损害训练的收敛性，因此，在每一轮训练中，都会积累较小的梯度值。然后，为了避免这些过时的梯度仅对训练产生很小的影响，可以应用动量校正，即设置折扣因子以校正残余梯度累积。

特别地，当训练大型DL模型时，交换对应的模型更新可能会消耗更多资源。使用在线版本的KD可以减少这种通信成本[159]。换句话说，交换模型输出，而不是交换每个设备上的更新模型参数，从而可以训练大型本地模型。除了通信成本外，还应考虑隐私问题。例如，在[160]中，可以通过利用训练有素的分类器的隐私泄漏来有目的地从训练数据中获得个人信息。在[161]中研究了边缘训练数据集的隐私保护。与[155] – [157]不同，在[161]的情况下，训练数据在边缘节点进行训练，并上传到云中以进行进一步的数据分析。因此，拉普拉斯噪声[162]被添加到这些可能暴露的训练数据中，以增强训练数据的隐私保证。

B. Federated Learning at Edge

在第VII-A节中，整体网络体系结构是明确分离的，特别是，培训仅限于在终端设备或边缘节点上而不是在它们两者之间进行。当然，通过这种方法，可以很容易地安排培训过程，因为不需要处理异构计算能力和端与端之间的网络环境。尽管如此，DL训练和DL推理都应该无处不在。联合学习（FL）[163]，[164]作为一种实用的DL训练机制在端部，边缘和云之间出现。尽管在本地FL的框架中，现代移动设备还是作为执行本地培训的客户。自然地，这些设备可以在边缘计算中得到更广泛的扩展[165]，[166]。云中的终端设备，边缘节点和服务器可以等效地视为FL中的客户端。假定这些客户端能够处理不同级别的DL训练任务，并因此将其更新贡献给全局DL模型。在本节中，将讨论FL的基础知识，改进和实际用例。

1) Vanilla Federated Learning: FL [163]，[164]无需上传数据进行中央云训练，就可以允许边缘设备使用自己收集的数据来训练其本地DL模型，而仅上传更新的模型。如图17所示，FL迭代地请求一组随机的边缘设备进行以下操作：1）从聚合服务器下载全局DL模型（以下使用“服务器”），2）将本地模型与已下载的全局模型一起训练自己的数据，并且3）仅将更新的模型上载到服务器以进行模型平均。通过将培训数据仅限制在设备端，可以显着降低隐私和安全风险，从而避免将培训数据上传到云中而导致[160]中所述的隐私问题。此外，FL引入了FederatedAveraging，将每台设备上的本地SGD与执行模型平均的服务器结合在一起。实验结果证实了FederatedAveraging对不平衡和非IID数据具有鲁棒性，并且可以简化训练过程，即减少训练DL模型所需的交流次数。

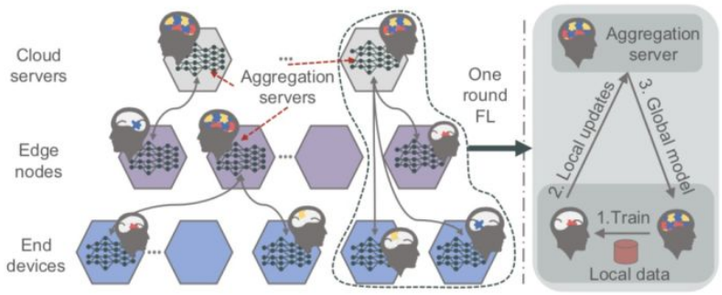

Fig. 17. Federated learning among hierarchical network architectures.

总而言之，FL可以应对边缘计算网络中的几个关键挑战：1）非IID训练数据。每个设备上的训练数据都是自己感知和收集的。因此，设备的任何单独的训练数据将无法代表全局数据。在FL中，这可以通过FederatedAveraging来满足； 2）沟通有限。设备可能会脱机或位于通讯环境差的地方。但是，在资源充足的设备上执行更多的训练计算可以减少全局模型训练所需的通信回合。此外，FL仅选择一部分设备在一个回合中上传其更新，因此成功处理了设备意外脱机的情况； 3）贡献不平衡。可以通过FederatedAveraging解决，具体来说，某些设备可能没有更多的FL自由资源，从而导致设备之间的训练数据和训练能力数量不尽相同； 4）隐私和安全。需要在FL中上传的数据只是更新的DL模型。此外，可以自然地应用安全聚合和差异隐私[162]，该安全聚合和差异隐私可用于避免公开包含在本地更新中的隐私敏感数据。

2) Federated Learning Optimization: 实际上，在FL中，不需要上载原始训练数据，从而大大降低了通信成本。但是，FL仍需要将本地更新的模型传输到中央服务器。假设DL模型的大小足够大，则将更新（例如模型的权重）从边缘设备上载到中央服务器也可能会消耗不可忽略的通信资源。为了应对这一挑战，[167]提出了结构化更新和草图更新，以增强客户端将更新上传到服务器时的通信效率。结构化更新意味着将模型更新限制为具有预先指定的结构，特别是1）低秩矩阵；或2）稀疏矩阵。另一方面，对于草绘的更新，将保留完整的模型更新，但是在将其发送到服务器进行全局模型聚合之前，将执行子采样，概率量化和结构化的随机轮换的组合操作以压缩完整的更新。

与仅研究减少上行链路的通信成本不同，[168]考虑了服务器到设备（下行链路）和设备到服务器（上行链路）的通信。对于下行链路，全局DL模型的权重被整形为向量，然后应用二次采样和量化[167]。自然，这种模型压缩是有损的，并且与上行链路（有多个边缘设备上载其模型以求平均）不同，无法通过在下行链路上求平均值来减轻这种损失。 Kashin的表示[169]可以在进行二次采样之前用作基础变换，以减轻后续压缩操作引起的误差。此外，对于上行链路，不需要每个边缘设备根据整个全局模型在本地训练模型，而仅将更新训练为较小的子模型。这些子模型是全局模型的子集，通过这种方式，减少了更新上传中的数据量。

同样，与云相比，边缘设备的计算资源也很少。还应考虑其他挑战：1）不仅通信，而且在边缘设备上的计算资源也受到限制； 2）在不同边缘设备上的训练数据集可能分布不均匀[170] – [172]。为了解决这些挑战，需要最优地确定所有参与设备之间在给定资源预算下的全局模型聚合的频率。基于推导的具有非IID数据分布的分布式学习的收敛边界，可以通过理论上的保证来优化聚集频率[170]。

为了加快FL中的全局聚合，[173]利用了无线计算[174]-[176]的优势，其原理是探索无线多址信道的叠加特性以计算所需的通过同时传输多个边缘设备来发挥作用。可以利用无线信道的干扰，而不仅仅是克服它们。在传输过程中，来自边缘设备的并发模拟信号可以自然地由信道系数加权，然后服务器只需要在聚合结果上叠加这些重整后的权重即可，而无需其他聚合操作。

3) Security Enhancement: 在香草FL中，本地数据样本在每个边缘设备上进行处理。这种方式可以防止设备向服务器泄露私有数据。但是，服务器也不应完全信任边缘设备，因为具有异常行为的设备会伪造或破坏其训练数据，这将导致毫无价值的模型更新，从而损害全局模型。为了使FL能够容忍对中毒数据集进行少量设备训练，鲁棒的联合优化[166]定义了修正的均值运算。通过不仅过滤掉中毒设备产生的值，还过滤掉正常设备中的自然异常值，可以实现保护全局模型免受数据中毒的鲁棒聚合。

除了有意攻击之外，还应关注不可预测的网络状况和计算能力对安全性造成的被动负面影响。 FL必须对意外退出边缘设备具有鲁棒性，否则，一旦设备断开连接，FL的同步将失败。为了解决这个问题，在[177]中提出了安全聚合协议，以达到容忍多达三分之一的设备无法及时处理本地训练或上传更新的鲁棒性。

反过来，FL中聚合服务器的故障可能会导致不正确的全局模型更新，从而扭曲所有本地模型更新。此外，边缘设备（具有大量数据样本）可能不太愿意与其他设备（具有较小的贡献）一起参与FL。因此，在[178]中，提出将Blockchain和FL结合为BlockFL来实现：1）在每个边缘设备而不是特定服务器上进行局部全局模型更新，确保设备故障在更新全局模型时不会影响其他局部更新； 2）刺激边缘设备参与FL的适当奖励机制。

VIII. DEEP LEARNING FOR OPTIMIZING EDGE

DNN（通用DL模型）可以提取潜在数据特征，而DRL可以通过与环境交互来学习处理决策问题。 边缘节点的计算和存储功能以及云的协作，使得使用DL优化边缘计算网络和系统成为可能。 关于各种边缘管理问题，例如边缘缓存，卸载，通信，安全保护等，

1）DNN可以处理网络中的用户信息和数据指标，以及感知无线环境和边缘状态 节点，并基于这些信息2）DRL可用于学习长期最佳资源管理和任务调度策略，从而实现边缘的智能管理，即表V中所示的智能边缘。

A. Adaptive Edge Caching Policy

从CDN（内容交付网络）[179]到在蜂窝网络中缓存内容，多年来已经对网络中的缓存进行了研究，以应对对多媒体服务的飞速增长的需求[180]。与将内容推送到用户附近的概念一致，边缘缓存[181]被认为是一种有前途的解决方案，可以进一步减少冗余数据传输，减轻云数据中心的压力并改善QoE。

应满足与边缘缓存有关的两个挑战：1）边缘节点覆盖范围内的内容流行度分布很难估计，因为它可能有所不同并且会随时空变化而变化[182]； 2）鉴于边缘计算环境中的大型异构设备，分层缓存体系结构和复杂的网络特性进一步困扰了内容缓存策略的设计[183]。具体而言，仅当已知内容流行度分布时才能推论出最佳边缘缓存策略。但是，用户对内容的偏爱实际上是未知的，因为它们的移动性，个人喜好和连接性可能一直在变化。在本节中，将讨论如图18所示的用于确定边缘缓存策略的DL。

1) Use Cases of DNNs: 传统的缓存方法通常需要很高的计算复杂度，因为它们需要大量的在线优化迭代来确定内容的放置和交付。但是，当使用DNN优化边缘缓存时，可以通过脱机训练来避免在线繁重的计算迭代，并且仅需要给出DL推理的优化策略即可。 DNN由用于数据正则化的编码器和后面的隐藏层组成，可以使用由最佳算法或启发式算法生成的解决方案进行训练，并部署以确定缓存策略[184]，从而避免了在线优化迭代。类似地，在[185]中，受关于部分缓存刷新的优化问题的输出具有某些模式的事实的启发，对MLP进行了训练，使其接受当前的内容流行度和最后的内容放置概率作为生成缓存刷新策略的输入。

如[184] [185]中所述，优化算法的复杂性可以转移到DNN的训练中，从而打破了使用它们的实际限制。在本文中，DL用于学习输入-解决方案关系。显然，基于DNN的方法仅在存在针对原始缓存问题的优化算法时才可用。因此，基于DNN的方法的性能受到固定优化算法的限制，不能被认为是自适应的。

另外，DL可以用于定制的边缘缓存[186]。具体来说，为了最大程度地减少无人驾驶汽车的内容下载延迟，将MLP部署在云中以预测要请求的内容的受欢迎程度，然后将MLP的输出传递到边缘节点（即RSU上的MEC服务器）。 [186]）。根据这些输出，每个边缘节点缓存最有可能被请求的内容。在自动驾驶汽车上，选择了CNN来预测车主的年龄和性别。一旦确定了所有者的这些特征，就可以使用k均值聚类[187]和二进制分类算法来确定哪些内容已经缓存在边缘节点中，应该进一步从边缘节点下载并缓存到汽车中。此外，关于充分利用用户的功能，[188]指出用户在不同环境下访问内容的意愿各不相同。受此启发，RNN可用于预测用户的轨迹。并且基于这些预测，然后可以预取用户感兴趣的所有内容并将其预先缓存在每个预测位置的边缘节点上。

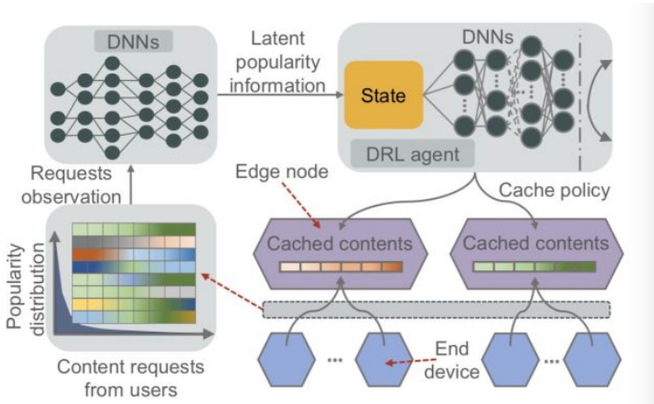

Fig. 18. DL and DRL for optimizing the edge caching policy.

2) Use Cases of DRL: 可以将第VIII-A1节中描述的DNN的功能视为整个边缘缓存解决方案的一部分，即DNN本身并不能解决整个优化问题。与这些基于DNN的边缘缓存不同，DRL可以利用用户和网络的上下文，并采用使长期缓存性能最大化的自适应策略[189]作为优化方法的主体。传统的RL算法受限于对手工功能的要求以及难以处理高维观测数据和动作的缺陷[190]。与与DL不相关的传统RL（例如Q学习[191]和MAB（多武装强盗）学习[182]）相比，DRL的优势在于DNN可以从原始观测数据中学习关键特征。结合了RL和DL的集成DRL代理可以直接从高维观测数据优化边缘计算网络中缓存管理方面的策略。

在[192]中，DDPG用于训练DRL代理，以最大化长期缓存命中率，从而做出适当的缓存替换决策。这项工作考虑了单个BS的情况，DRL代理决定是缓存请求的内容还是替换缓存的内容。在训练DRL代理时，奖励被设计为缓存命中率。此外，Wolpertinger体系结构[193]用于应对大型行动空间的挑战。详细地，首先为DRL代理设置主要操作集，然后使用K最近邻居（KNN）将实际操作输入映射到该操作集中的一个。通过这种方式，可以在不丢失最佳缓存策略的情况下故意缩小操作空间。与搜索整个操作空间的基于DQL的算法相比，具有DDPG和Wolpertinger架构的训练有素的DRL代理能够在降低运行时间的同时实现具有竞争力的缓存命中率。

B. Task Offloading Optimization

边缘计算允许边缘设备在能量，延迟，计算能力等约束下将其部分计算任务卸载到边缘节点[194]。如图19所示，这些约束提出了识别1）哪些边缘节点的挑战。应该接收任务； 2）边缘设备应分担多少比例的任务； 3）应该为这些任务分配多少资源。解决这种任务分流的问题是NP-hard [195]，因为至少需要对通信和计算资源以及边缘设备的竞争进行组合优化。特别是，优化应同时考虑随时间变化的无线环境（例如变化的信道质量）和任务卸载的要求，因此引起了使用学习方法的注意[196]-[205]。在所有与基于学习的优化方法相关的工作中，当多个边缘节点和无线电信道可用于计算分流时，基于DL的方法比其他方法更具优势。在这种背景下，整个卸载问题中的大状态空间和动作空间实际上使传统的学习算法[196] [206] [198]不可行。

1) Use Cases of DNNs: 在[199]中，负载问题的计算公式化为多标签分类问题。通过以离线方式穷举搜索解决方案，可以将获得的最佳解决方案用于以边缘计算网络的复合状态为输入，以卸载决策为输出的DNN训练。通过这种方式，可能不需要在线解决最佳解决方案，避免了迟来的卸载决策，并且可以将计算复杂性转移到DL训练中。

此外，在[202]中研究了关于区块链的特定卸载场景。边缘设备上挖掘任务的计算和能源消耗可能会限制区块链在边缘计算网络中的实际应用。当然，这些挖掘任务可以从边缘设备转移到边缘节点，但可能会导致边缘资源分配不公平。因此，所有可用资源都以拍卖的形式分配，以最大化边缘计算服务提供商（ECSP）的收入。基于最佳拍卖的分析解决方案，可以构建MLP [202]，并根据矿工（即边缘设备）的估值进行培训，以使ECSP的预期收入最大化。

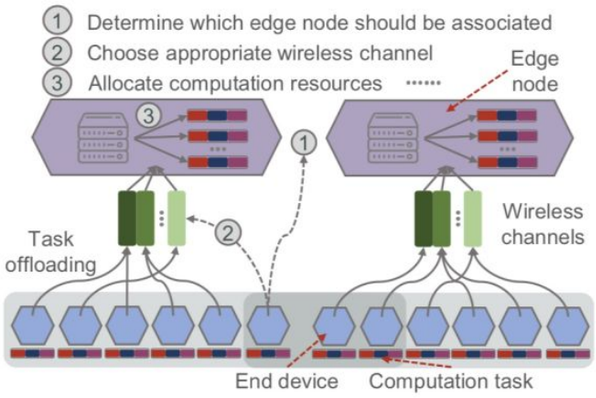

Fig. 19. Computation offloading problem in edge computing.

2) Use Cases of DRL: 尽管将计算任务卸载到边缘节点可以减少处理延迟，但是卸载的可靠性受到无线信道质量可能较低的困扰。对于任务卸载，不仅应关注延迟违反概率，而且还应关注解码错误概率。传输数据所用的编码速率对于使卸载满足所需的可靠性级别至关重要。因此，在[200]中，研究了编码块长度的影响，并通过DQL制定并解决了有关资源分配的MDP，以提高平均卸载可靠性。进一步探索调度边缘设备的细粒度计算资源，在[208]中，Double-DQL [74]用于选择最佳的动态电压和频率缩放（DVFS）算法。与DQL相比，实验结果表明Double-DQL可以节省更多能量并获得更高的训练效率。显然，随着边缘设备的增加，基于DQL的方法的操作空间可能会迅速增加。为了缩小动作空间的大小，可以在学习之前执行预分类步骤[203]。

在[201]，[204]中研究了由能量收集（EH）驱动的物联网边缘环境。在EH环境中，由于IoT边缘设备可以从周围的射频信号中收集能量，因此能量收集使卸载问题变得更加复杂。因此，CNN用于在学习过程中压缩状态空间[204]。此外，在[201]中，受奖励函数的加法结构的启发，在Double-DQL中应用了Q函数分解，从而改善了香草Double-DQL。但是，基于值的DRL仅能处理离散的操作空间。为了对本地执行和任务卸载执行更细粒度的电源控制，应考虑基于策略梯度的DRL。例如，DDPG可用于自适应地分配边缘设备的功率，它具有优于基于DQL的离散功率控制策略的优势[205]。

自由地让DRL代理接管整个计算卸载过程可能会导致巨大的计算复杂性。 因此，使用DRL进行部分决策可以大大降低复杂度。 在[209]中，首先将最大化加权总和的计算速率表述为问题的最终目标。 然后，将问题分解为两个子问题，即卸载决策和资源分配。 最后，仅通过使用DRL来解决NP困难的负载决策问题，就可以缩小DRL Agent的动作空间，并且也不会因为优化资源分配而影响卸载性能。

TABLE V
DL FOR OPTIMIZING EDGE APPLICATION SCENARIOS

		Related Work	DNNs or DRL	Related Methods	Objective
Adaptive Edge Caching Policy		[184]	DL	FCNN	Estimate content popularity and deduce proactive caching strategies
		[185]	DL	CNN	Avoid the repeated calculation of cache refreshing optimization
		[186]	DL	FCNN	Train caching policies in the self-driving car to minimize content downloading delay
		[188]	DL	RNN	Predict user mobility and cache contents of users interest in advance
		[207]	DRL	AC	Schedule users and content caching to minimize the average transmission delay
		[192]	DRL	DDPG	Maximize the long-term cache hit rate
Task Offloading Optimization		[199]	DL	FCNN	Optimize offloading actions to minimize the computation and offloading overhead
		[202]	DL	FCNN	Develop an optimal auction based on DL for the edge resource allocation
		[200]	DRL	DQL	Allocate computational resource for offloaded tasks in order to improve the average end-to-end reliability
		[208]	DRL	Double-DQL	Adaptively select dynamic voltage and frequency scaling algorithms for energy-efficient edge scheduling
		[203]	DRL	DQL	Determine the computation offloading decision and the allocated computation resource of all edge devices
		[205]	DRL	DDPG	Perform more fine-grained power control for local execution and task offloading
		[204]	DRL	DQL	Choose the MEC server and determine the offloading rate in energy harvesting environment
		[201]	DRL	Double-DQL	Optimize computation offloading policies for a virtual MEC system
		[209]	DRL	FCNN	Reduce the computational complexity without compromising the solution quality
Edge Management and Maintenance	Edge Communication	[210]	DL	RNN & LSTM	Assist the smooth transition of device connections and offloaded tasks between edge nodes
		[211]	DRL	DQL & TL	Control communication modes of edge devices and on-off states of their processors to minimize long-term system power consumption
	Edge Security	[212]	DRL	DQL	Provide secure offloading from the edge device to the edge node against jamming attacks
	Joint Edge Optimization	[213]	DRL	DQL	Improve the servicing performance of the edge computing network
		[207]	DRL	AC	Minimize the average end-to-end servicing delay
		[95]	DRL	Double-Dueling DQL	Improve the performance of future IoVs
		[97]	DRL	DQL	Optimize caching placement and computing resource allocation as well as to determine possible connections

C. Edge Management and Maintenance

如[214]中所实现的，边缘计算服务被设想为部署在蜂窝网络中的BS上。 因此，边缘管理和维护需要从多个角度（包括通信角度）进行优化。 许多工作着重于在无线通信中应用DL [215] – [217]。 但是，如本节中所讨论的，还应该考虑其他观点。

1) Edge Communication: 当边缘节点为移动设备（用户）提供服务时，应解决边缘计算网络中边缘设备的切换问题。基于DL的方法可用于协助边缘节点之间的连接平滑过渡[210]。具体而言，整个切换问题是使移动设备在其整个移动轨迹上从边缘节点移动到下一个节点的中断最小化。 MLP可用于预测给定位置和时间的可用边缘节点。此外，仍然需要评估移动设备与每个边缘节点之间的交互的成本（服务请求的等待时间），以确定移动设备应关联的最佳边缘节点。但是，对这些交互的成本进行建模需要更强大的学习模型。因此，实现了具有LSTM单元的两层堆叠RNN，以对交互成本进行建模。最后，基于预测可用边缘节点的能力以及相应的潜在成本，移动设备可以与最佳边缘节点相关联，因此将中断的可能性最小化。

为了解决通信场景中的能源最小化问题，需要采用多种模式（以服务于各种IoT服务），即C-RAN（云无线电接入网）模式，D2D（设备到设备）模式和FAP（雾）在无线接入点模式下，DQL可用于控制整个通信过程中边缘设备的通信模式以及处理器的开/关状态[211]。然后，当DQL代理确定给定边缘设备的通信模式和处理器的开-关状态时，整个问题将退化为RRH（远程无线电头端）传输功率最小化问题并得到解决。

2) Edge Security: 由于边缘设备通常仅配备有限的计算，能量和无线电资源，因此与边缘设备之间的传输相比于各种攻击（如干扰攻击和分布式拒绝服务（DDoS）攻击）更容易受到攻击。云计算。因此，应增强边缘体系结构的安全性。当然，安全保护通常需要额外的能量消耗以及计算和通信的开销。因此，每个边缘设备应优化其防御策略，即选择发射功率，信道和时间，而不会违反其资源限制。由于难以估计边缘计算网络的攻击模型和动态模型，因此优化面临挑战。

基于DRL的安全解决方案可以提供安全的卸载（从边缘设备到边缘节点）以防止干扰攻击[212]或保护用户位置隐私和使用模式隐私[218]。边缘设备观察边缘节点的状态和攻击特征，然后确定安全协议中的防御级别和关键参数。通过将奖励设置为抗干扰通信效率，例如信号的信噪比（SINR），接收到的消息的误码率（BER）和保护开销，可以训练基于DQL的安全代理以应对各种类型的攻击。

3) Joint Edge Optimization: 边缘计算可以满足智能设备的快速增长以及大量计算密集型和数据消耗型应用程序的出现。但是，这也使未来网络的操作更加复杂[219]。在全面资源优化方面管理复杂网络[16]尤其是在考虑未来网络的关键推动力的前提下，包括软

件定义网络（SDN）[220]，物联网，车联网（IoV）。

通常，SDN被设计用于将控制平面与数据平面分离，从而允许以全局视图在整个网络上进行操作。与边缘计算网络的分布式性质相比，SDN是一种集中式方法，将SDN直接应用于边缘计算网络具有挑战性。在[213]中，研究了适用于智慧城市的支持SDN的边缘计算网络。为了提高该原型网络的服务性能，在其控制平面中部署了DQL以协调网络，缓存和计算资源。

边缘计算可以为物联网提供更多计算密集型和对延迟敏感的服务，但也给有效管理以及存储，计算和通信资源的协同作用提出了挑战。为了最小化平均的端到端服务延迟，基于策略梯度的DRL与AC架构可以处理边缘节点的分配，是否存储请求内容的决策，边缘的选择执行计算任务和计算资源分配的节点[207]。 IoV是物联网的一种特殊情况，专注于互联车辆。与[207]中对集成网络，缓存和计算的考虑类似，具有更强大性能的Double-Dueling DQL（即，将Double DQL和Dueling DQL结合在一起）可用于编排可用资源，以提高性能。未来的IoV [95]。

此外，考虑到车辆在IoV中的机动性，可能很难打破硬服务期限约束，并且由于高度复杂性，这一挑战常常被忽略或未能充分解决。为了应对机动性挑战，在[97]中，首先将车辆的机动性建模为离散随机跳跃，并将时间维度划分为多个时期，每个时期包括几个时隙。然后，针对时隙的粒度，设计了一个小型的时标DQL模型，以根据精心设计的即时奖励功能来考虑车辆机动性的影响。最后，针对每个时间段提出了一个大型的时标DQL模型。通过使用这种多时标DRL，解决了有关移动性的直接影响和资源分配优化中难以忍受的大型动作空间的问题。