Edge-Cloud Polarization and Collaboration: A Comprehensive Survey for AI

Jiangchao Yao, Shengyu Zhang, Yang Yao, Feng Wang, Jianxin Ma, Jianwei Zhang, Yunfei Chu, Luo Ji, Kunyang Jia, Tao Shen, Anpeng Wu, Fengda Zhang, Ziqi Tan, Kun Kuang, Chao Wu, Fei Wu*, *Senior Member, IEEE,* Jingren Zhou, *Fellow, IEEE,* Hongxia Yang* *Member, IEEE,*

Abstract—Influenced by the great success of deep learning via cloud computing and the rapid development of edge chips, research in artificial intelligence (AI) has shifted to both of the computing paradigms, i.e., cloud computing and edge computing. In recent years, we have witnessed significant progress in developing more advanced AI models on cloud servers that surpass traditional deep learning models owing to model innovations (e.g., Transformers, Pretrained families), explosion of training data and soaring computing capabilities. However, edge computing, especially edge and cloud collaborative computing, are still in its infancy to announce their success due to the resource-constrained IoT scenarios with very limited algorithms deployed. In this survey, we conduct a systematic review for both cloud and edge AI. Specifically, we are the first to set up the collaborative learning mechanism for cloud and edge modeling with a thorough review of the architectures that enable such mechanism. We also discuss potentials and practical experiences of some on-going advanced edge AI topics including pretraining models, graph neural networks and reinforcement learning. Finally, we discuss the promising directions and challenges in this field.

Index Terms—Cloud AI, Edge AI, Edge-Cloud Collaboration, Hardware.

1 Introduction

Cloud computing concerns the provisioning of adequate resources to construct a cost-efficient computing paradigm for numerous applications [1] while edge computing can offer low-latency services. For cloud computing, it has flourished for a long period in the past decades and achieved a great success in the market, *e.g.,* Amazon EC2, Google Cloud and Microsoft Azure. According to the recent analysis [2], the global cloud computing market size was valued at USD 274.79 billion in 2020 and is expected to grow at a compound annual growth rate of 19.1% from 2021 to 2028. Concomitantly, Artificial Intelligence (AI), especially the compute-intensive Deep Learning [3], has enjoyed the tremendous development with the explosion of cloud computing. Nevertheless, the rapid increase of Internet of Things (IoT) [4] raises an inevitable issue of the data transfer from the edges to the data centers in an unprecedented volume. Specifically, about 850 ZB data is generated by IoT at the network edge by 2021, but the traffic from worldwide data centers only reaches 20.6 ZB [5]. This drives the emergence of the decentralized computing paradigm, edge computing, which turns out to be an efficient and well-recognised solution to reduce the transmission delay. Similarly in algorithmic application layer of edge computing, there is an urgent need to push the AI frontiers to the edges so as to fully unleash the potential of the modeling benefits [6].

The existing two computing paradigms, cloud computing and edge computing, polarize the algorithms of AI into different directions to fit their physical characteristics. For the former, the corresponding algorithms mainly focus on the model performance in generalization [7], robustness [8], fairness [9] and generation [10], [11] *etc.*, spanning from computer vision (CV), natural language process (NLP) to other industrial applications. To achieve better performances, a large amount of research from the perspectives of the data, the model, the loss and the optimizer is devoted to exploring the limit under the assumption of sufficient computing power and storage. For example, the impressive Generative Pre-trained Transformer 3 (GPT-3) [12] that has 175 billion parameters and is trained on the hundred-billion scale of data, can produce human-like texts. The AlphaFold [13] with the elaborate network design for the amino acid sequence is trained with a hundred of GPUs and makes the astounding breakthrough in highly accurate protein structure prediction. Nowadays, cloud computing is continuously advancing AI widespread to various scientific disciplines and impact our daily lives.

However, in terms of edge computing, it is still in its infancy to announce the success due to the constrained IoT resources. There are several critical limitations to design AI algorithms that run on IoT devices while maintaining the model accuracies. The most critical factor is the *processing speed* for the applicability of any edge application [14]. We usually use throughput and latency for the measurement, which respectively count the processing rate and characterize the time interval between a single input and its response. Second, *memory* like RAM and cache is the critical resource for building AI edge applications. Machine learning algorithms often take a significant portion of memory during

- *J. Yao, Y. Yao, F. Wang, J. Ma, J. Zhang, Y. Chu, L. Ji, K. Jia, J. Zhou and H. Yang are with DAMA Academy, Alibaba Group.*
  E-mail: {jiangchao.yjc, yy222244, wf135777, jason.mjx, zhangjianwei.zjw, fay.cyf, jiluo.lj, kunyang.jky, jingren.zhou, yang.yhx}@alibaba-inc.com
- *S. Zhang, T. Shen, A. Wu, F. Zhang, Z. Tan, K. Kuang, F. Wu and C. Wu are with Zhengjia University.*
  E-email: {sy_zhang, tao.shen, anpwu, fdzhang, tanziqi, kunkuang, chao.wu, wufei}@zju.edu.cn
- *Wu, F. and Yang, H. are corresponding authors.*

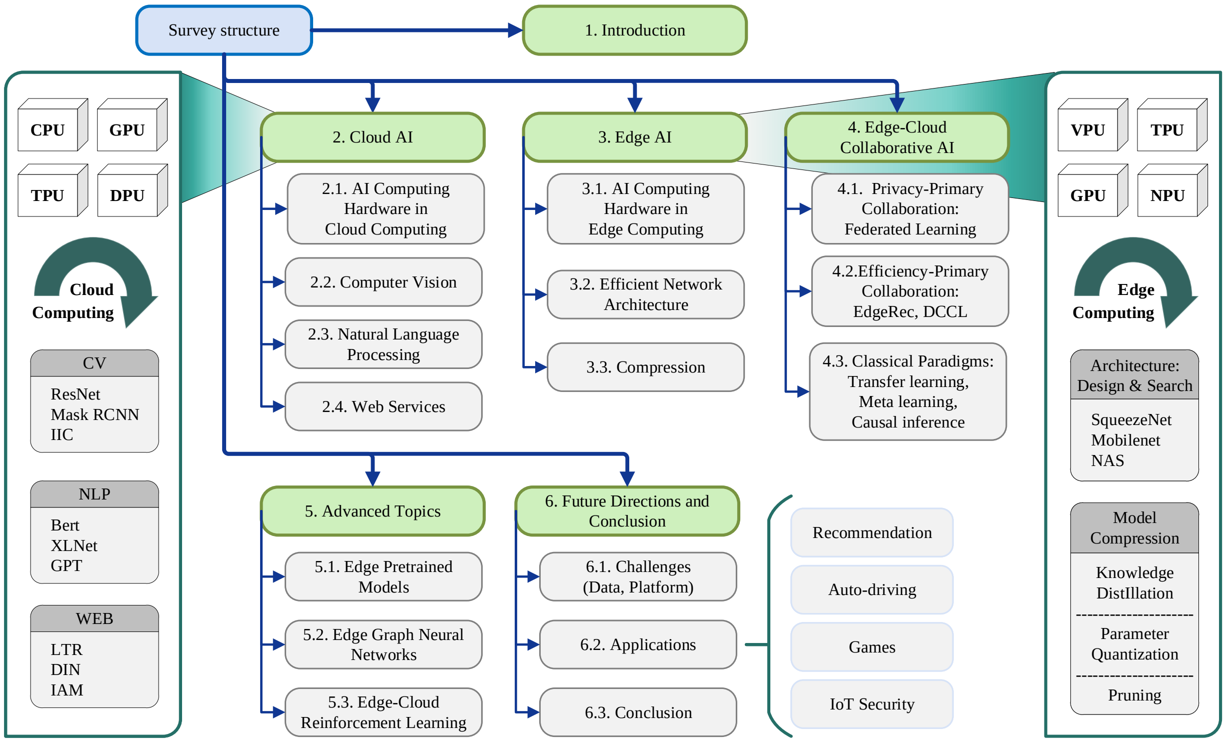

Survey structure
1. Introduction
2. Cloud AI
3. Edge AI
4. Edge-Cloud Collaborative AI
CPU
GPU
TPU
DPU
Cloud Computing
CV
ResNet
Mask RCNN
IIC
NLP
Bert
XLNet
GPT
WEB
LTR
DIN
IAM
2.1. AI Computing Hardware in Cloud Computing
2.2. Computer Vision
2.3. Natural Language Processing
2.4. Web Services
3.1. AI Computing Hardware in Edge Computing
3.2. Efficient Network Architecture
3.3. Compression
4.1. Privacy-Primary Collaboration: Federated Learning
4.2.Efficiency-Primary Collaboration: EdgeRec, DCCL
4.3. Classical Paradigms: Transfer learning, Meta learning, Causal inference
VPU
TPU
GPU
NPU
Edge Computing
Architecture: Design & Search
SqueezeNet
Mobilenet
NAS
Model Compression
Knowledge DistIllation
Parameter Quantization
Pruning
5. Advanced Topics
5.1. Edge Pretrained Models
5.2. Edge Graph Neural Networks
5.3. Edge-Cloud Reinforcement Learning
6. Future Directions and Conclusion
6.1. Challenges (Data, Platform)
6.2. Applications
6.3. Conclusion
Recommendation
Auto-driving
Games
IoT Security

Fig. 1: Layout of the Survey.

model building with the storage of model parameters and other auxiliary variables. Except the storage, querying the model parameters for processing and inference is both time-consuming and energy-intensive [15], yielding *power consumption* and *thermal performance* both are bottlenecks.

Given the above constraints, edge computing polarizes AI to a new research era, namely *edge intelligence* or *edge AI*[1] [16]–[18]. Contrary to entirely relying on the cloud, edge AI makes use of the edge resources to gain further AI insights. For example, one direction is to make the model lightweight. One representative work is MobileNet [19], which reduces the number of parameters from 29.3 million to 4.2 million and the number of computations by a factor of 8 while only losing 1% accuracy. Furthermore, EfficientNet [20] is proposed to systematically scale up CNN models in terms of network depth, width and input image resolution. It achieved the state-of-the-art performance via 8 times lesser memory but 6 times faster inference. Major enterprises, such as Google, Microsoft, Intel, IBM, Alibaba and Huawei, have put forth pilot projects to demonstrate the effectiveness of edge AI. These applications cover a wide spectrum of AI tasks, *e.g.*, live video analytics [21], cognitive assistance for agriculture [22], smart home [23] and the industrial IoT [17].

Despite distinct characteristics of cloud computing and edge computing, a complete real-world system usually involves their collaboration from the physical aspects to the algorithmic aspects. We term this cooperation as *Edge-Cloud collaboration*, and actually some early explorations have been made in both academia and industry, *e.g.*, federated Learning (FL) [24]. Recently, a successful industrial practice is the Taobao EdgeRec System [25]. In EdgeRec, the memory-consuming embedding matrices are deployed on the cloud and the lightweight component executes the real-time inference on the edge. Similarly in [26], a CloudCNN provides the soft supervision to each local EdgeCNN for the edge training and simultaneously, the EdgeCNN performs the real-time inference interacting with the input.

However, the huge diversity in different algorithmic areas prevents a systematic review from the complete edge-cloud scope. For example, the transformer [27], [28] conquers various benchmark completions in CV and NLP, but we did not witness its much progress on edge. Edge computing [6] has the quite promising IoT applications in the future intelligent life, but without the auxiliary from cloud computing, the powerful large models will be extravagant to most of us. Although FL [29] is elegant to protect the user privacy, given the heterogeneity and hardware constraints, we still have to explore some new solutions to remedy the performance degradation. All these concerns might be trapped in the local area and some works in the cross-field might enlighten us to find a better way. Motivated by this intuition, we conduct a comprehensive and concrete survey about the edge-cloud polarization and collaboration. Specifically, we are the first to set up the collaborative learning mechanism for cloud and edge modeling. To summarize, we organize the survey of this paper as follows and in Fig. 1:

- Section 2 gives an overview of cloud AI;
- Section 3 discusses edge AI;
- Section 4 reviews the architectures that enable the collaboration between cloud and edge AI;
- Section 5 discusses some on-going advanced topics, including pretraining models (PTM), graph neural networks (GNN) and reinforcement learning (RL) models that are

1. Edge computing originally refers to the computing in edge infrastructures like rounters. We here extend it to a broader range including smart phones *etc*. Edge AI therein we talk also includes on-device AI.

acronym	description
IoT	Internet of Things
CV	Computer Vision
NLP	Natural Language Processing
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
DNN	Deep Neural Network
GNN	Graph Neural Networks
RL	Reinforcement Learning
PTM	Pretraining Modeling
FL	Federated Learning
FRL	Federated Reinforcement Learning
KD	Knowledge Distillation
QA	Question Answering
NAS	Neural Architecture Search
PQ	Product Quantization

TABLE 1: Description of popular acronyms in our survey.

deployed on edge;
- Section 6 reviews the hardware development;
- We conclude the paper in Section 7 with some possible future directions and challenges.

2 Cloud AI

AI has achieved the tremendous development in the recent years, even outperformed the human performance in a range of open-source competition benchmarks [30], [31]. One main reason behind this success is the benefit from the cloud computing, *i.e.,* the large-scale distributed cluster, which greatly accelerate the training of AI models. We term them as *cloud AI*. Considering the real-world AI applications depend on the information carriers such as image and text, we review some computing hardwares and cloud models in the areas of CV, NLP and web services as exemplars.

2.1 AI Computing Hardware in Cloud Computing

There are many factors on hardware and software as well as the customer-friendly resource management technology (*e.g.,* the virtualization technology [32]–[34]) to thrive on the great success of cloud computing. We kindly refer readers to [1] for a complete analysis. In this part, we specially recap some computing hardware from the mainstream companies, which plays the crucial roles in the AI flourish.

2.1.1 CPU

The high performance of Central Processing Unit (CPU) is important to accelerate the computing-intensive AI tasks. In the last decade, there are many excellent breakthroughs in the processor design. As one exemplar in Intel, the Core i9-7920X CPU is the fastest processor for the general computation in the Core X-series processors. The competitor AMD released the Ryzen-series of products like Ryzen 5 2600 and Ryzen 9 3900X, both for desktops and IoTs based on the Zen micro-architecture. Recently, the AMD Ryzen Threadripper 3990X that has the best performance is the flagship of Ryzen series especially designed for the edge machine learning. We enumerate their release time of above CPU processors in Fig. 2 according to the time-varying website ranking[2] of Intel and AMD, but without any business purposes. Note that, the general CPU server is slow for the large-scale training of DNNs, and thus some specific speedup technologies from the task scheduling are usually deployed [35] or mixed with the following modules [36], [37].

2. https://benchmarks.ul.com/compare/best-cpus

2.1.2 GPU

Graphics Processing Unit (GPU) was originally designed to create images for computer graphics and video games. In early 2010, researchers found that GPUs can be used to accelerate calculations of DNNs, based on which AlexNet achieved a great success in the ImageNet challenge [38]. The GPU behind AlexNet is Nvidia GeForce GTX 580, which is a high-end graphics card launched by NVIDIA in 2010. Subsequently, NVIDIA released the Tesla K80 and Tesla P100 that further improved the computation performance. Recently, Tesla V100 as well as Tesla A100 again catches the eyes due to the impressive PTM [39]–[41] in language understanding and image generation tasks. We similarly include the release time of some popular GPUs in Fig. 2 according to the time-varying open ranking[3].

2.1.3 TPU and DPU

Tensor Processing Unit (TPU) is a dedicated integrated circuit developed and used in Google Cloud, which is superior in the DNN computation. Its first generation is identified by the high-bandwidth loop [42] and the second generation is known by faster cost-effective mixed precision training [43]. The third generation provides significant performance benefits over TPUv2 and fits better in the larger-scale network architecture. In Google I/O 2021, Google launches the latest TPUv4 with nearly two times performance over TPUv3. Data Processsing Unit (DPU) is a recently proposed programmable specialized electronic circuit for data-centric computing, which frees the computing load of CPU or GPU in the hardware aspect[4]. It is practically useful for the acceleration of AI models in face of the expensive loading and offloading cost of large-scale neural networks like PTM [12].

2.2 Computer Vision

Among various research areas, CV is a longstanding field, which asks computers to derive the semantics from digital images, videos, and other visual inputs. In this part, we review some classical tasks, *i.e.,*, Image Recognition, Object Detection and Image Segmentation.

Image recognition involves analyzing images and identifying objects, actions, and other elements in order to draw conclusions. In the 1880s, the predecessor of CNN "neocognitron" was proposed. Then, Lecun *et al.,* [44] proposed the CNN operator and successfully applied it to handwritten digital character recognition, with an error rate of less than 1%. In 2012, AlexNet [38] won the ImageNet competition [45] and incorporated many enticing techniques such as ReLU nonlinearity, dropout, and data augmentation. Following AlexNet, deeper structures like VGG-16 [46] and GoogleNet [47] were explored. To better handle gradient vanishing issue, ResNet [48] used a skip-connection scheme

3. https://analyticsindiamag.com/top-10-gpus-for-deep-learning-in-2021/
4. https://blogs.nvidia.com/blog/2020/05/20/whats-a-dpu-data-processing-unit/

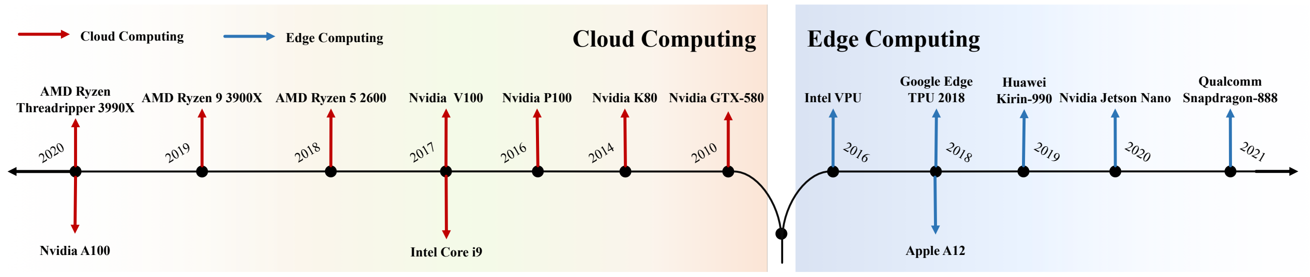


Fig. 2: Some exemplar computing hardwares widely used by the algorithmic development of AI models in cloud computing and edge computing, collected according to the open websites as shown in the paragraphs without any business purposes.

to form a residual learning framework. As such, it scales convolutional layers up to 152 and becomes the winner of multiple vision tasks in ILSVRC 2015 and MS COCO 2015.

Object detection identifies and locates objects in an image/scene. In recent years, one-stage and two-stage object detectors [49]–[52] have achieved noticeable improvements. However, these methods rely on deep convolution operations to learn intensive features, resulting in a sharp increase in the cost of computing resources and an apparent decrease in detection speed [53]. Therefore, how to address these problems and enable real-time detection becomes an important line of research in object detection. Recently, pruning techniques and knowledge distillation are effective tools to build a lightweight object detector [54], [55].

Image segmentation [56] is another fascinating technique that separates an image into several recognizable segments. Apparently, image segmentation requires tremendous human effort in labeling pixels. Recently, IIC [57] uses the mutual information-based clustering to output a semantic segmentation probability map in image pixels. AC identifies probabilities of pixels over categories by an autoregressive model and maximizes mutual information across two different "orderings" [58], [59]. In addition to generic domains, many experts also focus on investigating the domain-specific image segmentation techniques [60]–[62].

Except above subareas, great success also has been made in other CV tasks such as super-resolution [63], image restoration [64], and image generation [65]. For example, DALL-E [65] trains a 12-billion-parameter autoregressive transformer for zero-shot text-to-image generation, and achievse the significant generalization.

2.3 Natural Language Processing

With a long-term development, NLP has been developed into a broad sub-areas including tagging, named entity recognition, question answering and machine translations *etc*. In the following, we briefly review three recently challenging sub-areas. For other pos tagging or text categorization tasks, we kindly refer to these literatures [66], [67].

Machine translation (MT) [68] has achieved great success in movie caption translation, sport games and international political conference, which saves the human labor and domain knowledge in translation. Past works can be roughly divided into rule-based [69], statistical-based [70] and neural machine translations [71]. The early attempt in neural MT is recurrent continuous translation model [72], which leverages an autoregressive mechanism to automatically capture

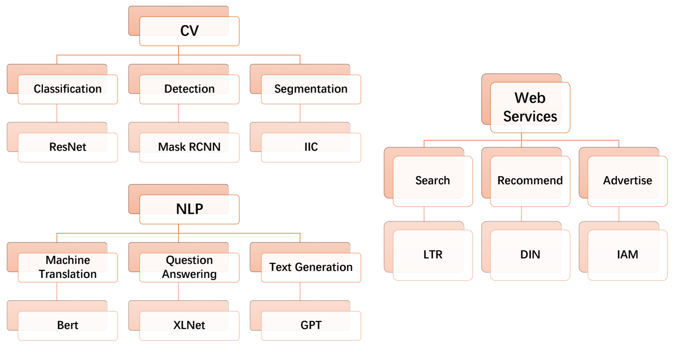


Fig. 3: We enumerate the representative methods at the bottom nodes of some subareas in CV, NLP and Web Services.

word ordering, syntax, and meaning of the source sentence explicitly. Sutskever *et al.*, [73] improves it via an encoding-decoding approach of the bidirectional LSTM to learn the long-term dependencies. The following variants mainly target to design the more efficient attention modules. Recently, the BERT-based neural architecture achieves the new state-of-the-art on a range of MT benchmarks by enlarging the capacity with the large-scale corpora [74].

Question Answering (QA) [75] as a classical technique is commonly depolyed in some well-known engines like Microsoft Windows and Apple Siri. The industrial QA usually consists of multiple stages *e.g.*, query recognition and expansion, answer selection and fine-grained ranking spanning from text, image and videos [76]. The mainstream research falls into modeling the QA pairs and the negative sampling, exploring the implicit matching between multiple objects and the answer [77]. Dialogue system can be considered as a more complex QA, which generates the answer in the multiple rounds [78]. Compared to the single retrieval, the interactive sentence generation is more critical.

Text generation is an emerging direction along with the development of the large-scale PTMs such as GPT series in the recent years [39], [79]. Several generation tasks like poetry generation and story generation have shown impressive performances. In order to make the generation more human-like, the GAN-style [80] and VAEs-style [81] have been respectively explored in PTMs to leverage their potential merits. Except totally structure-free generation, some knowledge-enhanced text generation that considers the text hint, constraint-aware and graph-based knowledge have been investigated [82], [83]. Besides, some cross-domain text generation like Visual QA and reading comprehension are also explored to understand the multi-modalities [39], [40].

2.4 Web Services

Recommendation, Search and E-advertisement has been the successful business paradigms in web services [84]. The corresponding Cloud AI models as the core components of these paradigms are widely explored in many enterprises including Google, Amazon and Microsoft [85].

2.4.1 Search

Web search [86] retrieves the object of interest from a huge number of candidates according to the human query. The keyword search has been investigated almost along with the emergence of world wide web [87]. More challenges raise when query is the image or the video. In the image search, the model should extract any possible instances from the image to match the candidates and thus many works focus on enhancing the feature extraction ability [88]. Considering the domain bias and noise, some approaches proposed to pretrain the model on the in-domain clean subset and then finetune on the open-domain dataset [89]. Due to the latency constraint for the multimedia data, the hash techniques are usually explored to accelerate the retrieval [90].

2.4.2 Recommendation

Recommender systems have been widely studied in the last decade and become an indispensable infrastructure of web services [91]. The related recommendation methods are progressively improved with the development of collaborative filtering, deep learning and sequential modeling. The early stage mainly focuses on the user-based collaborative filtering [92], the item-based collaborative filtering [93] and matrix factorization [94]. As deep learning achieved a great success, several variants of collaborative filtering combined with DNNs were proposed [95]–[100]. They leverage DNNs to activate high-level semantics for more accurate recommendation. Sequential modeling as another perspective to model the user interests has been successfully applied to recommender systems [101]. With the architecture evolving, several methods based on GRU [102], Attention [103]–[108] have achieved remarkable performances.

2.4.3 Advertisement

Advertisement we review here refers to the computational advertising developed for web services [109]. It is an intersection of multiple disciplines like Advertising, Marketing and Computer Science. Computational Advertising builds on the deep understanding of the web data and user preference, and at the same time taking the cost and the revenue into account. A range of works study the recommendation system for advertising [110], guaranteed target display advertising [111] in a dynamic way or real-time bidding [112] by analyzing the contextual data. Except the cost-effective impression, most algorithms target to find the best strategic planing among the complex markets with the uncertain revenue, or explore the optimal advertising path by inferring the causal relations with the incentive bonus [113]. With the development of AI, models combined with advertising have drawn more attention. It constructs an automatic optimization for the complex marketing environment while still maintaining the advertisement efficiency compared to the traditional advertisement [114].

3 Edge AI

On the mobile platform, the breakthrough of AI has spawn a wealth of intelligent applications, such as virtual assistants [115] and personalized recommendation [116]. The traditional cloud-based paradigm requires the data uploading, which may violate the user privacy and heavily depend on the network conditions [117]. The way to alleviate the above problems is edge inference, where we can place models partially or fully on mobile devices and make predictions locally. However, edge inference is highly nontrivial as the computing, storage and energy resources of mobile devices are limited, and many research efforts have been devoted. In this section, we review some representative hardware and important techniques to ease models on the edges.

3.1 AI Computing Hardware in Edge Computing

The demand for edge AI grows rapidly due to the issues of bandwidth, privacy or compute-transmission balance. Many hardwares have been developed to meet the requirements of numerous edge AI applications including the light-weighted servers such as Raspberry Pi and Nvidia TX 2, and some function-oriented hardwares. Due to the space limitation, we simply enumerate some representative types of the function-oriented hardwares in the following and more general servers for edge computing can refer to [118].

3.1.1 VPU

Vision processing unit (VPU) is an emerging class of microprocessor used in edge AI. It allows the efficient execution of edge vision workloads and achieves a balance between power supply efficiency and computing performance. One of the most popular examples is the Intel Neural Compute Stick, which is based on the Intel Movidius Myriad X VPU. This plug-in and play device can be easily attached to edges running Linux, Windows, Raspbian, including Raspberry Pi and Intel NUC *etc*. In terms of machine learning framework it supports TensorFlow, Caffe, MXNet and PyTorch *etc*.

3.1.2 Edge TPU

Google has built Edge Tensor Processing Unit (Edge TPU)[5] to accelerate the ML inference on edges. It is capable of running the classical compressed CNNs such as MobileNets, MobileNets SSD and Inception as well as TensorFlow Lite models, and has been applied into the real-world detection and segmentation. Currently, it is limited to open for public.

3.1.3 Mobile GPU

Enterprises like Nvidia and Qualcomm explore to integrate the GPU to accelerate the computation on edges. The representative is NVIDIA Jetson Nano[6], which includes an integrated 128-core Maxwell GPU, quad-core ARM A57 64-bit CPU, 4GB LPDDR4 memory, along with support for

5. https://cloud.google.com/edge-tpu
6. https://developer.nvidia.com/embedded/jetson-nano

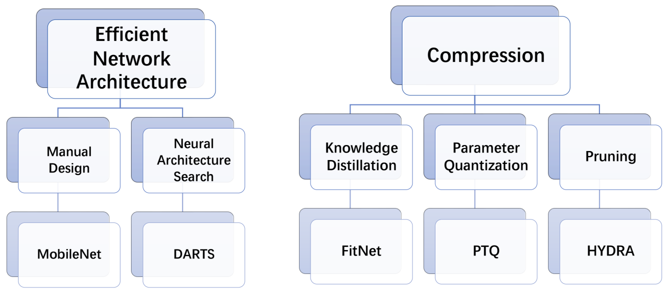


Fig. 4: We enumerate some representation methods in the area of Efficient Network Architecture and Compression.

MIPI CSI-2 and PCIe Gen2 high-speed I/O. Apple also developed the processor A12 Bionic for iPhone and iPad, which is a 64-bit ARM-based system on a Apple chip. It includes a dedicated neural network hardware, which Apple calls the "next generation neural engine". Similarly, HiSilicon released Kirin 990 5G, a 64-bit high-performance mobile ARM 5G SoC. More recently, Qualcomm announced the Snapdragon 888 on 2020, which combines the computing capabilities of the new Hexagon 780 and GPU, providing the impressive 26 TOPS computing power.

3.1.4 NPU

Neural Processing Unit (NPU) is to find an efficient configuration to control a large number of resources [119], [120]. Typically, DNNs require a large amount of data for training and ask for large memory. To resolve the need for off-/on-chip memory bandwidth, NPUs usually rely on data reuse and unnecessary computations skipping. For example, ARM Ethos N-77[7] delivers up to 4 TOPS of performance, scaling to 100s of TOPS in multicore deployments. It supports machine learning frameworks like TensorFlow, TensorFlow Lite, Caffe2, PyTorch, MXNet, and ONNX. Tensorflow and Pytorch also have generic support for deploying models on Android NPU via Google NNAPI.

3.2 Efficient Network Architecture

3.2.1 Manual Design

There are a lot of works exploring the lightweight network architectures. One representative is SqueezeNet [121], which leverages the iteration of a squeeze layer and a expansion layer for the parameter compression. MobileNet [19] decomposes the conventional convolution into the composition of a depth-wise convolution and a point-wise convolution. The intuition behind MobileNet is that the low rank merit of the convolution kernel makes the decomposition approximately equivalent, and thus computation can be accelerated by a two-stage convolution. Similarly, MobileNet-v2 [122] shows that the high dimensional features can actually be expressed through compacting low-dimensional features. Then, they proposes a new layer "inverted residual with linear bottleneck" to reduce the parameters. ShuffleNet [123] shows the point-wise convolution in MobileNet is expensive when the input dimension is high. It leverages the group convolution together with the channel shuffle to reduce the computational cost and the parameter space.

7. https://www.arm.com/products/silicon-ip-cpu/ethos/ethos-n77

3.2.2 Neural Architecture Search

Compared to the manual design, neural architecture search (NAS) enables us to automatically explore the efficient network architectures. The classical NAS [124] utilizes RNN as the controller to generate a sub-network, and then perform the training and evaluation, and finally updates the parameters of the controller. There are two major challenges in NAS [124]–[126]. The first comes from the non-differential objective [127]–[129]. Specifically, the performance of the sub-network is non-differential, which makes it infeasible to optimize the controller directly. Fortunately, the strategic gradient methods in RL can be a surrogate to update the controller parameters. Another challenge relates to the computationally expensive pipeline where we have to train each sub-network updated by the controller from scratch [130]–[132]. Towards this end, efficient neural architecture search introduces the weight sharing technique and greatly reduces the search time [133]. Recently, there is a rapidly growing trend in this direction, which can be divided in the perspectives of search space [131], search policy sampling network [134] and the performance-aware selection [132].

3.3 Compression

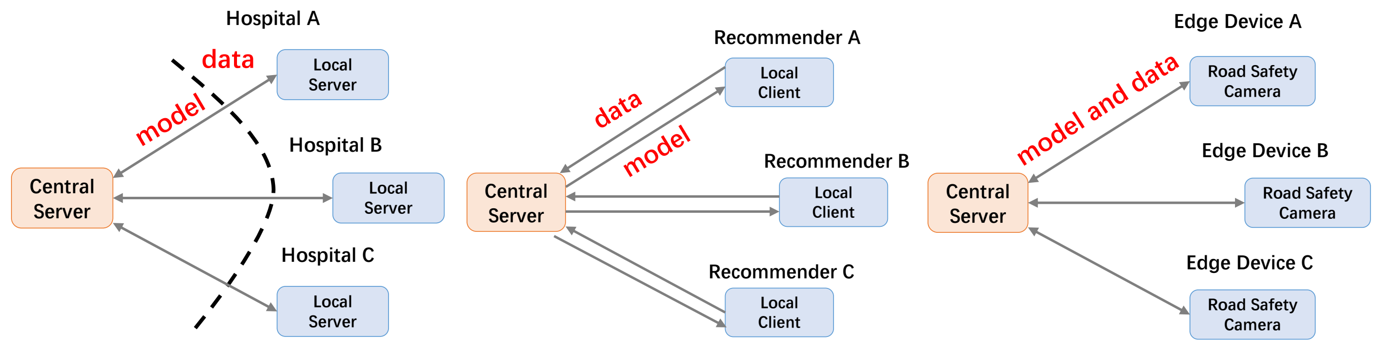
3.3.1 Knowledge Distillation

KD [135] is widely used to transfer the knowledge from complex models or model ensembles to the lightweight models. According to the distillation manner, KD can be divided into response-based KD [135]–[138], feature-based KD [139], [140] and relation-based KD [141], [142].

- **Response-Based KD.** It takes the network output as the soft target to teach the student model. It is simple but effective for model compression, and has been widely used in different tasks and applications.
- **Feature-Based KD.** DNNs are good at learning multiple levels of feature representation by abstraction. Therefore, the output of the intermediate layer, *i.e.*,, the feature map, can be used as the knowledge to supervise the training of the student model. FitNet [139] improved the training of the student model by matching the feature map between teachers and students directly. Subsequently, a range of other methods have been proposed to follow this paradigm [140], [143]–[148]. For example, Zagoruyko *et al.*, [140] derived an attention map from the original feature maps to express knowledge.
- **Relation-Based KD.** This lines of methods explore the relationship between different layers or data samples for distillation. Specifically, to explore the relationships between different feature maps, a flow of solution process (FSP) is proposed, which is defined by the gram matrix between two layers [141]. The FSP matrix summarizes the relations between pairs of feature maps. It is calculated using the inner products between features from two layers. To use the knowledge from multiple teachers, two graphs are formed by respectively using the logits and features of each teacher model as the nodes [149], [150].

3.3.2 Parameter Quantization

Quantization has shown a great success in both training and inference phases of DNN models [151], [152]. Specifically,



(a) Privacy-Primary Collaboration (b) Efficiency-Primary Collaboration (I) (c) Efficiency-Primary Collaboration (II)

Fig. 5: For each prototype of Edge-Cloud Collaborative AI mentioned in the paragraph, we present an instance. Specifically, in privacy-primary collaboration (a) *e.g.*, federated learning, the patient data in the hospital is very private, and thus during training and serving, only the model parameters are allowed to communicate. In efficiency-primary collaboration (I) *e.g.*, EdgeRec, the fine-grained user behavior data is collected to the central server, which trains on-device recommendation (personalized) models and deploys to clients. Finally, in efficiency-primary collaboration (II), both central server and local clients will respectively maintain independent models which interact with each other over data and model features.

the breakthrough of the half-precision and mixed-precision training [151], [153]–[159] has enabled an order of magnitude higher throughput in AI accelerators. However, it has proven very difficult to go below half-precision without significant tuning, and thus most of the recent research has focused on Parameter Quantization (PQ) [152]. Currently, the PQ methods can be divided into Quantization-Aware Training and Post-Training Quantization.

- **Quantization-Aware Training (QAT):** In QAT, the forward pass and the backward pass are performed on the quantized model in floating point, and the model parameters are quantized after each gradient update. Performing the backward pass with floating point is important, since accumulating the gradients in quantized precision can result in the zero gradient or gradients that have high error, especially in low-precision [160]–[165]. A popular approach to address this is to approximate the gradient by the Straight Through Estimator (STE) [166]. However, the computational cost of QAT is usually very high, since the retraining of the model may take hundreds of epochs, especially for the low bit quantization.
- **Post-Training Quantization (PTQ):** An alternative is Post-Training Quantization, which performs quantization and adjustment of the weights without fine-tuning [167]–[175]. The overhead of PTQ is often negligible, making PTQ appliable in situations where data is limited or unlabeled. However, this often comes at the cost of the lower accuracy compared to QAT, especially for the low-precision quantization. In PTQ, all the quantization parameters are determined without any re-training of the DNN model.

3.3.3 *Pruning*

Pruning is another way to reduce the parameter space by removing some computation paths in the model. Previous methods in this direction can be categorized into two main categories, one-time pruning and runtime pruning [176]. Specifically, for the former, there are three lines. One line of methods focus on pruning after model training [177]–[180]. It aims to design the certain criteria like value magnitudes [179] and second-derivative information [177] to remove the least salient parameters. Another line of methods [181]–[185] focus on jointly learning the sparse structures with model parameters by considering different downstream hardware features, *e.g.*, OFA [186]. These works explore the efficient strategies to avoid the expensive prune-retrain cycles for diverse constraints. The third line aims to prune networks in initialization, which saves the training resources [187]–[189]. The Lottery Ticket Hypothesis [190] demonstrates that the dense network contain sub-networks, that can reach a comparable accuracy with the full network.

Runtime pruning focus on the dynamic path selection to enable the real-time inference with the limited computation budgets. For example, Huang *et al.*, [191] implements the early-exit mechanism by injecting a cascade of intermediate classifiers throughout the deep CNNs. The layers are evaluated one-by-one with the forward process and the early-stop will execute once the CPU time is depleted, namely the latter layers are pruned. Yu *et al.*, [192] introduces a slimmable neural network, which adjusts its width on the fly based on the resource constraints. In contrast to the depth pruning, reducing width helps reduce the memory footprint in inference. Besides, some others re-consider the input-dependent assumption, since treating all inputs equally may cause unnecessary resource consumption [191], [193]–[197]. For inputs that are easy to distinguish, a simpler model might be sufficient. The pioneering works use handcrafted control decisions [191], [193]. For instance, Huang *et al.*, [191] terminates the inference once the intermediate classifiers output the confidence score exceeding a pre-determined threshold. The subsequent improvements propose to learn a network selection system, which adaptively prunes the full network for each input example. For example, Bolukbasi *et al.*, [194] propose an adaptive early-exit strategy by introducing extra classifiers to determine whether the current example should proceed to the next layer. Some others [195]–[197] utilize RL to learn the dynamic pruning decisions, which has a higher structure variability compared to the early-exit mechanism.

4 Edge-Cloud Collaborative AI

Edge-cloud collaborative modeling has received significant interest across separate research communities due to the recent privacy concerns, the latency requirements or the personalization [6]. For example, for the patient data, it is private and sensitive to open for the training and serving of AI models. This motivates the development of federated learning, which distributes the training to the local server for the purpose of privacy and aggregates the final model in the cloud server. From this perspective, federated learning is a classical edge-cloud "collaboration" that makes AI more practical. Actually, there are some similar exemplars about the different forms of collaboration and to sum all, we can categorize them into two sets: privacy-primary collaboration and efficiency-primary collaboration (*cf.* Figure 5).

4.1 Privacy-Primary Collaboration: Federated Learning

As aforementioned above, the collaboration of FL is privacy-primary in the sense that raw data of each client is stored locally and will not be exchanged or transferred [29]. Typically, FL can be roughly divided into two lines of works, *i.e.*, *cross-device* and *cross-silo*, respectively. In cross-device FL, there are substantial devices like phones, laptops, or IoT. In cross-silo FL, differently, there are organizations where data silos naturally exist. Another categorization given by [198] divides FL into horizontal FL, vertical FL and federated transfer learning, according to how data of different participants overlaps. In modern FL frameworks, there are several challenges that hinder effective edge-cloud collaboration. In this paper, we focus on three major challenges, *i.e.*, data heterogeneity, communication efficiency and attacks on FL.

4.1.1 Statistical Heterogeneity

The real-world data samples are not usually independent and identically distributed over numerous devices [199], resulting in heterogeneous edge models and gradients. Typically, there are five sources of data heterogeneity: 1) *Feature skew*: The marginal distribution of features may differ between edges. 2) *Label skew*: The marginal distribution of labels may differ between edges. 3) *Same label, different features*: The conditional distribution of features given labels may differ between edges. 4) *Same features, different labels*: The conditional distribution of labels given features may differ between edges. For example, the symbol ✓ represents `correct` in many countries and `incorrect` in some others (*e.g.*, Japan); and 5) *Quantity skew*: Clients can store drastically varying volumes of data. Unfortunately, real-world FL datasets present a mix of these phenomenons, and it is an urgent need to drive measures to alleviate statistical heterogeneity for edge-cloud collaboration. In this nascent research area, FL personalization addresses this issue by introducing a two-stage collaboration framework. The first stage is to collaboratively build a global model, followed by a customization stage for each customer with the client's private data [200]. Recently, several methods have been proposed to achieve personalization, enhanced by transfer learning, meta training, and KD techniques. A learning-theoretic framework with generalization guarantees is presented by [201]. Wang et al. [202] construct different personalization strategies, including the model graph, and the training hyperparameters for different edges. Jiang et al. [203] build connections between FL and meta-learning, and interpret FedAvg as a popular algorithm, Reptile [204]. Shen et al. [205] presents an FL method based on KD and transfer learning that allows clients to train their own model independently with the local private data. Instead of training one global model, Masour et al. [201] consider device clustering and learn one model per cluster.

4.1.2 Communication Efficiency

Communication cost is an important factor when deploying FL to the real-world scenarios given the limited communication budget. Many algorithms [206]–[208] have explored to improve the communication efficiency by accelerating the convergence rate of the training model or introducing a hierarchical counterpart of federated learning. Specially, the latter perspective is scalably in many practical cases by introducing the intermediate servers to perform the partial aggregation. The subsequent improvement regarding the dynamic resource allocation and the task management to save the transmission cost in this hierarchy are also explored [209], [210]. In the future, with the development of the unmanned aerial vehicles, it is possible to achieve the near-instant super-connectivity for federated learning in a more communication-efficient manner [211], [212].

4.1.3 Attacks on FL

Current FL protocol designs are usually vulnerable to attackers inside and outside of the system, yielding the privacy and the robustness at risk. There are two serious threats to FL privacy and robustness [219]: 1) poisoning attacks against robustness; and 2) inference attacks against privacy.

The impact of *poisoning attacks* on the FL model is determined by the extent to which the backdoor players participate in the assaults, as well as the amount of training data poisoned. Model poisoning attacks seek to prevent global model learning [220] or hide a backdoor trigger into the global model [221]. These attacks contaminate the changes of local models before these changes are uploaded to the server. The Byzantine attack [220] is a form of untargeted model poisoning attack that uploads arbitrary and harmful model changes to the server that fools the global model.

Though FL protects data privacy by shielding off local data from being directly accessed, FL still suffers from privacy risks due to *inference attacks*. For example, Deep Leakage from Gradient [222] presents an optimization approach that can retrieve the raw images and texts critical for model improvement. Existing studies in privacy-preserving FL are often built on traditional privacy-preserving approaches, including: (1) *homomorphic encryption* [223]; (2) *Secure Multiparty Computation*, [224]; and (3) *differential privacy* [225]. To protect against an honest-but-curious opponent, Hardy et al. [226] used FL on data encrypted by the homomorphic technique. Yao et al. [227] introduces a collaborative computation on their inputs without disclosing them to one another. Given the resource limits of mobile devices, it is expected that privacy-protection solutions must be computationally inexpensive, communication-efficient and resistant to failure. Truex et al. [228] present a FL system under the protection of the local differential privacy framework, while minimizing the overwhelming impact of noise.

TABLE 2: Device-Cloud Collaborative Modeling

	Collaboration Manner		**Collaboration Concerns**			
	co-training	co-inference	privacy	storage efficiency	communication efficiency	personalization
Semantic QA cache [213]		✓			✓	✓
EdgeRec System [25]		✓			✓	✓
Auto-Split [214]		✓		✓	✓	
CoEdge [215]		✓		✓	✓	
Colla [216]	✓			✓		✓
DCCL [217]	✓			✓		✓
MC^2-SF [218]	✓	✓				✓
FedAvg [24]	✓		✓		✓	
FML [205]	✓		✓	✓	✓	✓
Personalized FedAvg [203]	✓		✓		✓	✓
HyperCluster [201]	✓		✓		✓	✓
Federated Evaluation [202]	✓	✓	✓			✓

4.2 Efficiency-Primary Collaboration

Although FL is a popular framework that considers the data privacy and governance issue, some real-world applications may not be sensitive to the privacy but care more about the factors like communication budget or personalization. In the recent years, there are some emerging edge-cloud collaborations in this direction and we term this paradigm as the *efficiency-primary collaboration* (*cf.* Figure 5b-5c).

4.2.1 Split-deployment

The direct collaboration is the partially inference offloading from cloud to edge or from edge to cloud [229]–[232], which separates one model into two components, one on the cloud side and the other placed on the edge side [233], [234]. We term it as the *split-deployment*. One successful practice is from the Taobao EdgeRec System [25], where the memory-consuming embedding matrices encoding the attributes are deployed on the cloud side and the lightweight component executes the remaining inference on the edge side. Amin *et al.*, [214] introduced an Auto-Split solution to automatically split DNNs models into two parts respectively for the edge and for the cloud. The coded edge computing based on the auction mechanism [235], [236] is also a promising method to achieve the efficient resource allocation considering the edge capacities. To optimize the latency, [237] [238] leverage lossy data compression techniques to reduce the transmission data size. Some scenarios involve in the unmanned aerial vehicles as the assisted edge-cloud computing, which requires a more complete joint learning paradigm for the task allocation and management [239].

4.2.2 Edge-Centralized Personalization

Another type of edge-cloud collaboration is to leverage the decentralized advantage of the edge side in personalization by setting up an auxiliary cloud model. For example, Lu *et.al.*, [216] proposes a collaborative learning method, COLLA, for the user location prediction, which builds a personalized model for each device, and allows the cloud and edges learned collectively. In COLLA, the cloud model acts as a global aggregator distilling knowledge from multiple edge models. Ding *et.al.*, [26] introduces a collaborative framework, where a CloudCNN provides the soft supervision to each local EdgeCNN for the edge training and simultaneously, the EdgeCNN performs the real-time inference interacting with the vision input. Yao *et.al.*, [217] proposes a edge-cloud collaboration framework for recommendation via the backbone-patch decomposition. It greatly reduces the computational burden of edges and guarantees the personalization by introducing a MetaPatch mechanism and re-calibrates the backbone to avoid the local optimum via MoModistill. Extensive experiments demonstrate its superiority on the recommendation benchmark datasets.

4.2.3 Bidirectional Collaboration

One more intensive type of edge-cloud collaboration could be bidirectional, where we consider the independent modeling on each side and they maintain the interactive feedback to each other during both the training and serving. One recent exploration is a Slow-Fast Learning mechanism for Edge-Cloud collaborative recommendation developed on Alibaba Luoxi Platform [218]. In this method, the slow component (the cloud model) helps the fast component (the edge model) make predictions by delivering the auxiliary latent representations; and conversely, the fast component transfers the feedbacks from the real-time exposed items to the slow component, which helps better capture user interests. The intuition resembles the role of System I and System II in the human recognition [240], where System II makes the slow changes but conducts the comprehensive reasoning along with the circumstances, and System I perceives fast to make the accurate recognition [241]. The interaction between System I and System II allows the prior/privileged information exchanged in time to collaboratively meet the requirements of the environment. With the hardware advancement of mobile phones, IoT devices and edge servers, it will be meaningful to pay attention to such a bidirectional collaborative mechanism in different levels [6].

4.3 Rethinking Collaboration in Classical Paradigms

Edge-cloud collaborative learning can be formulated as a two-stage optimization problem, where we train a model on one side (cloud or edge) and then further optimize it on the other side (edge or cloud). Considering this, we rethink this problem from the perspectives of transfer learning, meta-learning, and causal inference.

4.3.1 Transfer Learning.

In edge-cloud collaborative learning, the data distribution naturally differs from edge to edge and from edge to cloud. Towards this end, heterogeneous transfer learning [242]–[249] would greatly improve the bidirectional model adaptation between the edges and the cloud. There are roughly two lines of heterogeneous transfer learning works addressing the difference in feature space, *i.e.*, symmetric transformation, and asymmetric transformation. Symmetric transformation [243], [248], [250] aims to learn domain-invariant representations across different domains. [251] addressed unsupervised transfer learning, which indicates a mostly labeled source domain with no target domain labels. [247] proposed Cross-Domain Landmark Selection as a semi-supervised solution. As a counterpart, by asymmetric transformation mapping [252]–[255], the feature space of the source domain is aligned with that of the target domain. A semi-supervised method of adapting heterogeneous domains was proposed, called Semi-Supervised Kernel Matching Domain Adaptation [255]. [256] learns an enhanced feature space by jointly minimizing the information loss and maximizing the domain distribution alignment.

4.3.2 Meta-learning.

Meta-learning is another successful knowledge transfer framework. Different from transfer learning where models learn from solving the tasks in the source domain, meta-learning expects that the model learns how to quickly solve new tasks. Based on meta-learning, on-cloud training (meta-training) could yield models that can learn quickly in heterogeneous edge environments (meta-learning). Recently, [257] proposed to employ the spiking neural networks and meta-learning with streaming data, permitting fast edge adaptation. Based on MAML [258], MELO [259] is another work that learns to quickly adapt to new mobile edge computing tasks. Another major challenge in edge-cloud collaborative learning is the limited computation capacity of edges. By consolidating both meta-learning and model compression, existing researches learn light-weight models that can quickly adapt to the edge environments [260]–[263]. [261] proposes an end-to-end framework to seek layer-wise compression with meta-learning. [262] learns a meta-teacher that is generalizable across domains and guides the student model to solve domain-specific tasks.

4.3.3 Causal Inference.

There is a substantial and rapidly-growing research literature studying causality [264], [265] for bias reduction and achieving fairness. Causal theory is essential to edge-cloud collaborative learning for the following two reasons: 1) on-cloud training is supposed to yield a generalizable model, which is free from confounding effects and model bias, *w.r.t* the heterogeneous edge data distributions; 2) edge training should avoid over-fitting plagued by spurious correlations between the input and the outcome. [266] builds connections among model compression, causal inference, and out-of-distribution generalization. With the causal effects as the basis, they propose to make decisions on model components pruning for better generalization. More works dive into the intersection of causality and out-of-domain generalization [267]–[274]. [270] proposes to reserve semantics that is discriminative in the target domain by embracing disentangled causal mechanisms and deconfounding. [273] assumes the relationship between causal features and the class is robust across domains, and adopts the Markov Blanket [275] for the causal feature selection. [271] proposes a causal regularizer to recover the causation between predictors and outcome variables for stale prediction across unknown distributions. [272] designs an instrumental variable based methods for achieving invariant relationship between predictors and outcome variable for domain generalization.

5 Advanced Topics

5.1 Edge Pretraining Models

PTM, also known as the *foundation model*s [276], such as BERT [27] and GPT-3 [277], have become an indispensable component of modern AI, due to their versatility and transferability especially in few-shot or even zero-shot settings. The scale of the foundation models have been growing tremendously recently [40], [41], [278]–[280], as researchers observe that larger models trained on larger corpora generally leads to superior performance on downstream tasks [27], [277]. The large scale, which can range from millions of to trillions of parameters, however, raises a critical question as to whether edge intelligence can enjoy the merits of the increasingly powerful foundation models. On the other hand, the unique challenges faced by edge agents, such as the heterogeneity of the deployment environments and the need of small sample adaptation, constitute an ideal testbed for demonstrating the versatility and transferability of the foundation models. We herein discuss three crucial directions that require exploration before we can unleash the power of the foundation models on edges.

5.1.1 Model Compression

Compressing a large foundation model into a small one is necessary due to the storage and/or network bandwidth constraint of many edge mobiles such as modern mobile phones. General techniques discussed in previous sections, such as model quantization and weight pruning, can be readily adopted, while there are also techniques specifically tailored for the Transformer architecture commonly adopted by the foundation models. For instance, parameter sharing across layers has been proven effective by ALBERT [281]. KD [135] remains the most popular solutions. DistilBERT [282] uses a subset of the layers of the teacher model to form the student for distillation. TinyBERT [283] improves upon the vanilla logit-based distillation method [135] by adding extra losses that align the immediate states between the teacher and the student. MobileBERT [284] introduces bottleneck layers to reduce the dense layers' parameters while keeping the projected hidden states' dimension unchanged, for convenient feature map transfer and attention transfer. MiniLM [285] demonstrates that it is effective to distill the dot products between the attention queries, keys, and values, which do not impose constraints on the hidden size or the number of layers of the student. So far the distilled tiny models usually comes with around 10M parameters, and it is unclear whether this size can be further reduced without much performance loss.

5.1.2 Inference Acceleration

Some techniques for reducing the model size, for example reducing the hidden sizes or the number of layers, can

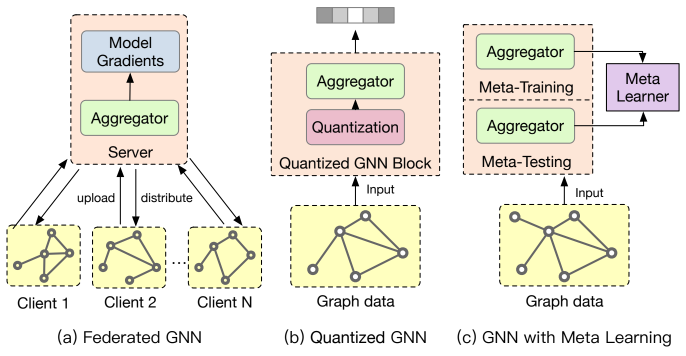


Fig. 6: The ongoing evolution of Edge GNNs. (a) Federated GNN, (b) Quantized GNN, (c) GNN with Meta Learning.

also bring the faster inference. Yet, as a foundation model typically has tens of layers, an interesting question arises: whether it is necessary to use all the layers during inference, since there may be simple samples or easy downstream tasks that do not necessitate using the full model. PABEE [286] demonstrates that *early exiting*, i.e. dynamically stopping inference once the intermediate predictions of the internal classifier layers remain unchanged for a number of steps, is indeed effective with BERT. However, it remains unclear whether early exiting is equally applicable to generative models such as GPT-3 and text generation tasks.

5.1.3 Few-sample and Few-parameter Adaptation

The traditional method is to fine-tune a PTM on each target downstream task's samples. However, fine-tuning typically involves updating almost all parameters of a foundation model and is not effective enough in terms of sample efficiency [287]. It can lead waste of storage and network bandwidth since each fine-tuned model requires independent resources. Moreover, an edge is typically a few-sample learning environment, for example, only consisting of data associated with one single mobile phone. We thus need to seek a more efficient approach in place of fine-tuning for few-sample and few-parameter adaptation. The recent emergence of prompt-based learning [288] and instruction tuning [287] represent promising directions. These recent methods mine natural language templates for the downstream tasks and use the templates to form input sentences. With a powerful generative language model, such templates can guide the model to output the correct predictions for the downstream tasks, where the model prediction is in the form of natural language as well. These recent paradigms require no parameter updating and can even achieve zero-shot learning in some cases [287]. However, so far prompt-based learning and instruction tuning focus mostly on large-scale models, not the type of tiny models for edges.

5.2 Edge Graph Neural Networks

In recent years, Graph Neural Networks have achieved the state-of-the-art on graph-structured data in various industrial domains [289], including CV [290], [291], NLP [292], [293], traffic [294], [295], recommender systems [296] and chemistry [297]. It can learn high-level embedding from node features and adjacent relationship, thus effectively deal with graph-based tasks. The rapid growth of node feature and their adjacent information drive the success of GNN, but also pose challenges in integrating GNN into the edge-cloud collaboration framework, like data isolation, memory consuming, limited samples and generalization*etc*. Recently, some efforts have emerged to address the above problems from aspects of FL, quantization and meta learning. In this part, we briefly review these works.

5.2.1 Federated GNN

To collaborate the graph data distributed on different edges to train a high-quality graph model, recent researchers have made some progress in FL on GNN [298]–[311]. The key idea of FL is to leave the data on the edges and train a shared global model by uploading and aggregating the local updates, *i.e.*, model parameters, to a central server.

For example, Feddy [298] proposes a distributed and secure framework to learn the object representations from multi-device graph sequences in surveillance systems. AS-FGNN [300] further proposes a separated-federated GNN model, which decouples the training of GNN into two parts: the message passing part that is done by clients separately, and the loss computing part that is learnt by clients federally. FedGNN [302] applies the federated GNN to the task of privacy-preserving recommendation. It can collectively train GNN models from decentralized user data and meanwhile exploit high-order user-item interaction information with privacy well protected. FedSage [311] studies a more challenging yet realistic case where cross-subgraph edges are totally missing, and designs a missing neighbor generator with the corresponding local and federated training processes. FL-AGCNs [304] considers the NAS of GNN for the FL scenarios with distributed and private data.

To alleviate the heterogeneity in graph data, some works, e.g., FedCG [305] and GCFL [310], leverage clustering to reduce statistical heterogeneity by identifying homogeneous, while FedGL [306] exploits the global self-supervision information. SpreadGNN [307] extends federated multi-task learning to realistic serverless settings for GNNs, and utilizes a novel optimization algorithm with a convergence guarantee to solve decentralized multi-task learning problems. In addition to data, graphs can also exist as relationships among clients. For example, CNFGNN [308] leverages the underlying graph structure of decentralized data by proposing a cross-node federated GNN. It bridges the gap between modeling complex spatio-temporal data and decentralized data processing by enabling GNN in FL.

5.2.2 Quantized GNN

To systematically reduce the GNN memory consumption, GNN-tailored quantization that converts a full-precision GNN to a quantized or binarized GNN can emerge as a solution for resource-constrained edge devices [312]–[316]. SGQuant [312] proposes a multi-granularity quantization featured with component-wise, topology-aware, and layer-wise quantization to intelligently compress the GNN features while minimizing the accuracy drop. Degree-Quant [313] performs quantization-aware training on graphs, which results in INT8 models often performing as well as their FP32 counterparts. BGN [314] learns binarized parameters and enables GNNs to learn discrete embedding.

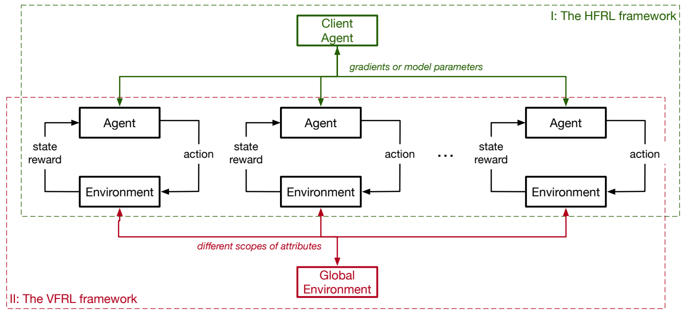


Fig. 7: The Federated RL framework. I: the horizontal FRL framework. II: the vertical FRL framework. Each agent-environment inner loop can be deployed on edge, and the client agent or global environment could be cloud-based, making the entire system as edge-cloud collaborative.

Bi-GCN [316] binarizes both the network parameters and the node attributes and can significantly reduce the memory consumptions by 30x for both the network parameters and node attributes, and accelerate the inference by about 47x.

5.2.3 GNN with Meta Learning

Recently, several meta learning methods to train GNNs have been proposed to solve the limited samples problem [317]. Most of the existing works [318]–[323] adopt the Model-Agnostic Meta-Learning algorithm [258]. Its outer loop updates the shared parameter, whereas its inner loop updates the task-specific parameter for the current task. G-META [318] uses local subgraphs to transfer subgraph-specific information and learn transferable knowledge faster via meta gradients with only a handful of nodes or edges in the new task. AMM-GNN [319] proposes a graph meta-learning framework for attributed graphs, which leverages an attribute-level attention mechanism to capture the distinct information of each task and thus learns more effective transferable knowledge for meta-learning. MI-GNN [320] studies the problem of inductive node classification across graphs and proposes a meta-inductive framework to customize the inductive model to each graph under a meta-learning paradigm.

5.3 Edge-Cloud Reinforcement Learning

Reinforcement Learning (RL) is a special machine learning manner by learning a policy through the interaction between agent and environment, which is close to the human learning way. RL could provide special functionalities such as train-and-error and long-term optimization which is often neglected by traditional unsupervised and supervised learning methods [324]. In classical RL studies, it is generally assumed that the agent is an edge device itself or governed by a cloud-based policy [325]. On the other hand, an RL system trained and interacted with the cloud-edge architecture has only been discussed in a limited scope. Nevertheless, there are some thorough investigations for some special domains, which are summarized as follows.

5.3.1 Federated Reinforcement Learning

There are some works which investigate FL [326] within the RL scope [327]–[331]. As summarized in [332], the idea of FRL could be divided into two main categories: horizontal FRL (HFRL) and vertical FRL (VFRL). Figure 7 illustrates the general frameworks, including HFRL with the server-client architecture, and VFRL with each edge-based environment as a part of the global environment.

In HFRL, the mobile devices distribute geographically but face similar tasks. The well-studied topic distributed RL is close to HFRL, and HFRL can be viewed as a security-enhanced distributed RL. For example, [328] studies the non-i.i.d. issue of data by deliberately choosing reasonable participating edges during model update. [329] proposes a two-timescale (fast and slow) DQN framework and FL is adopted in the fast timescale level and is trained in a distributed manner. [330] proposes an effective fault tolerance framework given the existence of failed workers. In VFRL, edge devices belong to the same group but their feature spaces are different. VFRL is relatively less studied than HFRL by so far; FedRL [327] is an example which builds a shared value network and uses the Gaussian differential on information shared by edges. Besides that, Multi-agent RL shares a lot similarities with VFRL; yet VFRL requires modeling on the partially observable Markov decision process while MARL usually assume full observability of system.

There are also other FRL studies which do not belong to either HFRL or VFRL. For example, [333] proposes Federated Reward Shaping in which reward shaping is employed to share federated information of agents. The Multi-task FRL (MT-FedRL) [334] achieves federation of policy gradient RL between agents by smoothing average weight on the agents' parameters. Both works are built on the server-client architecture.

5.3.2 RL-Assisted Optimization

There are also substantial studies which use RL as an aside system in cooperation with mobile-cloud system, to deal with optimization issues including online resource allocation [335], task scheduling [336], workload scheduling [337], computation offloading [338]–[344], and service mitigation [345]. Among these works, applications are implemented on the Internet of Things [335], [336], [338], [341], 5G network [340], telemonitoring [344], or vehicular terminals [343]. The basic motivation is that those optimization problems are generally NP-hard and easier to solve by DRL built on MDP [341], [342]. The detailed RL methods applied in these attempts include Q-Learning [337], [344], DQN [335], [338], REINFORCE [336], DDPG [339], PPO [343], and meta-RL [341]. Some key elements, such as storage space [342] or context impact [344], might also be took into account and are used to determined the computation stage of service .

6 Future Directions and Conclusion

6.1 Challenges

Although Edge-Cloud collaborative learning is a promising paradigm to a broad real-world applications, there are several challenges unsolved hindering its development. We summarize the main concerns in the following.

6.1.1 Data

To our best knowledge, the edge-cloud collaboration open-source datasets are very scarce, which limits the explore in the academic area. The only dataset[8] is released for the Mobile Edge Intelligence in recommendation. The reasons for the real-world open-source datasets are two-fold. On one hand, the fine-grained features on the edge side like the user real-time in-app scrolling, are usually not transmitted to the cloud due to the communication cost and the instant serving bottleneck. That is to say, we cannot completely understand the characteristics of data on the edge side by only training the model on the current cloud-based datasets. On the other hand, for the interactive scenarios like Luoxi [218], it requires the specific data collection to guarantee the distribution consistency between the training and the test. Simply simulation on the cloud-based dataset by decomposition cannot recover the real-world scenarios. Therefore, more open-source benchmark datasets contributed to this area will promote the research and industrial development.

6.1.2 Platform

The software platform is critical to explore the edge-cloud collaboration, since it is expensive to construct a collaboration environment and simulate the heterogeneous edges and the communication noise. However, it is still in its vacancy to build the well-established platforms that are friendly to a range of algorithmic study. For example, in FL, we have to handle the uncontrollable number of local models uploaded from the edge devices in the real-world scenarios, which may affect the convergence of the training [29]. Besides, a systematic analysis on the model training, deployment and evaluation are still lack, which is critical to measure the methods. The current open platforms mainly considering the algorithmic implementation like Luoxi[9]. In the future, it will be quite useful to establish the fully functional platform for the open comparison from both academia and industry.

6.2 Applications

6.2.1 Recommendation

Recommendation systems enable users to find and explore information easily and become increasingly important in a wide range of online applications such as e-commerce, micro-video portals, and social media sites. Despite the huge success, modern recommender systems still suffer from the user-oriented bias/fairness issue [346], [347], privacy leakage [348], [349] and high-latency response [350]. For example, centralized on-cloud training will be inevitably biased towards some privileged users, such as active users, resulting in an enlarged performance gap or unfairness among users. Embracing edge-cloud collaborative learning [217] opens up possibilities to address these issues. Expressly, edge training (cloud $\rightarrow$ edge) permits personalization that is free from biased collaborative filtering. Edge models can fully leverage edge features and provide low-latency services, such as dynamic interest modeling, and re-rank. As a counterpart, upon edge models, on-cloud training (edge $\rightarrow$ cloud) benefits from privileged distillation without access to sensitive user data, achieving both privacy protection and full personalization.

8. https://tianchi.aliyun.com/dataset/dataDetail?dataId=109858
9. https://github.com/luoxi-model/luoxi_models

6.2.2 Auto-driving

Essentially, a vehicle entirely controlled by machines without any human input will possess the acclaimed banner of being autonomous. Nowadays, an important aspect of self-driving vehicles integrating with the cloud is the capability of using the over-the-air electronic communications [351]. As well as various sensors for detecting the outside world, the vehicle will be equipped with on-board computer processors and electronic-oriented memory technology, and can communicate with the cloud via a communication device [352]. Besides data communication, edge-cloud collaborative learning techniques will further enhance autonomous driving on safety, functionality, and privacy [353]. Initially trained on clouds, edge machine learning models (cloud $\rightarrow$ edge) can interpret real-time raw data, make decisions based on the derived insights, and learn from the feedback from real-time road conditions. Real-time model adaptation is essential for safety and efficiency improvement, and accidents and traffic congestion reduction. Raw data like driving records might contain sensitive contents. Edge models can hereby enhance the centralized training with privacy.

6.2.3 Games

Video games have soared as one of the most popular ways to spend time. To ensure a seamless gaming experience, existing games struggle to figure out the work division of cloud computing and edge computing [354]–[356]. With edge-cloud collaborative learning, we can leverage the advantages of both sides. On the edge side, we can do more personalization and responsive actions that are sensitive to latency. For example, in sandbox games where the gameplay element is to give players a great degree of creativity and freedom on task completion and Non-Player-Character interaction, personalized AI for the dialogue generation and action-taking are an enticing element. Edge training permits such personalization by taking player behaviors, decisions, and preferences as input. Meanwhile, we can maintain the interactive feedback of edges and the cloud during both training and inference by transferring latent representations or leaving highly intensive but latency-insensitive training tasks for the cloud. Such a bidirectional collaboration brings more creation and freedom to users, ensures a seamless user experience, and opens up new possibilities for advanced gaming systems, such as the Metaverse [357], [358].

6.2.4 IoT Security

Internet of Things security refers to the protection of edges and networks from malicious attacks [359]. Most existing IoT systems are vulnerable to attacks [360], where threats in IoT include (but are not limited to) lack of proper data encryption and malicious software. Currently, some promising explorations [361], [362] have been developed for the secure resource management in the blockchain networks, which is an important future direction in the context of edge/cloud computing. However, in a larger landscape, it still requires more consideration about the safety trade-off in the edge-cloud collaboration. On the one hand, collaborative learning transfers model parameters, latent representations, and back-propagated gradients between edge-edge and edge-cloud, avoiding the communication attack on transferred sensitive data. On the other hand, in bidirectional collaboration, a trustable cloud model can identify malicious edges and mitigate their negative effects, which can be computationally intensive and typically not affordable on edges.

6.3 Conclusion

The rapid development of computing power drives AI to flourish, and evolve into three paradigms: cloud AI, edge AI and edge-cloud collaborative AI. To comprehensively understand the underlying polarization and collaboration of various paradigms, we systematically review the advancement of each direction and build a complete scope. Specially, our survey covers a broad areas including CV, NLP and web services powered by cloud computing, and simultaneously discusses the architecture design and compression techniques that are critical to edge AI. More importantly, we point out the potential collaboration types ranging from privacy-primary collaboration such as federated learning to efficient-primary collaboration for personalization. We rethink some classical paradigms in the perspective of collaboration that might be extended into edge-cloud collaboration. Some ongoing advanced topics for edge-cloud collaboration are also covered. Finally, we summarize the milestone products of cloud computing and edge computing in the recent years, and present future challenges and applications.

7 Acknowledgement

This project is partially supported by NSFC No. U20A20222.

References

[1] T. L. Duc, R. G. Leiva, P. Casari, and P.-O. Östberg, "Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey," *ACM Computing Surveys (CSUR)*, 2019.
[2] M. V. M. Size, "Share & trends analysis report," *URL: https://www. grandviewresearch. com/industry-analysis/cloud-computing-industry Accessed*, 2020.
[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, 2015.
[4] K. Ashton *et al.*, "That 'internet of things' thing," *RFID journal*, 2009.
[5] C. V. Networking, "Cisco global cloud index: Forecast and methodology, 2016–2021," *White paper. Cisco Public, San Jose*, 2016.
[6] 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*, 2019.
[7] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory," in *Summer school on machine learning*, 2003.
[8] L. P. Hansen and T. J. Sargent, *Robustness*. Princeton university press, 2011.
[9] S. Barocas, M. Hardt, and A. Narayanan, "Fairness in machine learning."
[10] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," in *ICML*, 2015.
[11] W. Fedus, I. Goodfellow, and A. M. Dai, "Maskgan: Better text generation via filling in the _," in *ICLR*, 2018.
[12] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.
[13] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, 2021.
[14] J. Nielsen, *Usability engineering*. Morgan Kaufmann, 1994.
[15] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, 2017.
[16] 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*, 2019.
[17] L. Li, K. Ota, and M. Dong, "Deep learning for smart industry: Efficient manufacture inspection system with fog computing," *IEEE Transactions on Industrial Informatics*, 2018.
[18] F. Jeronimo, "Mobile ai and the future of intelligent devices," *2017 White Paper IDC*, 2017.
[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," in *ICLR*, 2017.
[20] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019.
[21] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, 2017.
[22] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *MobiSys*, 2014.
[23] J. Cao, L. Xu, R. Abdallah, and W. Shi, "Edgeos_h: A home operating system for internet of everything," in *ICDCS*, 2017.
[24] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.
[25] Y. Gong, Z. Jiang, Y. Feng, B. Hu, K. Zhao, Q. Liu, and W. Ou, "Edgerec: recommender system on edge in mobile taobao," in *CIKM*, 2020.
[26] C. Ding, A. Zhou, Y. Liu, R. Chang, C.-H. Hsu, and S. Wang, "A cloud-edge collaboration framework for cognitive service," *IEEE TCC*, 2020.
[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019.
[28] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *arXiv preprint arXiv:2101.01169*, 2021.
[29] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, 2021.
[30] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[31] K. Grace, J. Salvatier, A. Dafoe, B. Zhang, and O. Evans, "When will ai exceed human performance? evidence from ai experts," *Journal of Artificial Intelligence Research*, 2018.
[32] F. Lombardi and R. Di Pietro, "Secure virtualization for cloud computing," *Journal of network and computer applications*, vol. 34, no. 4, pp. 1113–1122, 2011.
[33] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, 2013.
[34] J. Tang, J. Nie, Z. Xiong, J. Zhao, Y. Zhang, and D. Niyato, "Slicing-based reliable resource orchestration for secure software defined edge-cloud computing systems," *IEEE Internet of Things Journal*, 2021.
[35] A. A. Awan, H. Subramoni, and D. K. Panda, "An in-depth performance characterization of cpu-and gpu-based dnn training on modern architectures," in *Proceedings of the Machine Learning on HPC Environments*, 2017, pp. 1–8.
[36] Y. Jiang, Y. Zhu, C. Lan, B. Yi, Y. Cui, and C. Guo, "A unified architecture for accelerating distributed {DNN} training in heterogeneous {GPU/CPU} clusters," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 463–479.
[37] A. Jain, N. Alnaasan, A. Shafi, H. Subramoni, and D. K. Panda, "Optimizing distributed dnn training using cpus and bluefield-2 dpus," *IEEE Micro*, 2021.
[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, 2017.
[39] L. Floridi and M. Chiriatti, "Gpt-3: Its nature, scope, limits, and consequences," *Minds and Machines*, 2020.

[40] J. Lin, R. Men, A. Yang, C. Zhou, M. Ding, Y. Zhang, P. Wang, A. Wang, L. Jiang, X. Jia, J. Zhang, J. Zhang, X. Zou, Z. Li, X. Deng, J. Liu, J. Xue, H. Zhou, J. Ma, J. Yu, Y. Li, W. Lin, J. Zhou, J. Tang, and H. Yang, "M6: A chinese multimodal pretrainer," *arXiv preprint arXiv:2103.00823*, 2021.

[41] A. Yang, J. Lin, R. Men, C. Zhou, L. Jiang, X. Jia, A. Wang, J. Zhang, J. Wang, Y. Li, D. Zhang, W. Lin, L. Qu, J. Zhou, and H. Yang, "Exploring sparse expert models and beyond," *arXiv preprint arXiv:2105.15082*, 2021.

[42] T. Norrie, N. Patil, D. H. Yoon, G. Kurian, S. Li, J. Laudon, C. Young, N. P. Jouppi, and D. A. Patterson, "The design process for google's training chips: Tpuv2 and tpuv3." *IEEE Micro*, 2021.

[43] Y. Wang, G.-Y. Wei, and D. Brooks, "Benchmarking tpu, gpu, and cpu platforms for deep learning." *CoRR*, 2019.

[44] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, 1989.

[45] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[49] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *ECCV*, 2018.

[50] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017.

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[52] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *CVPR*, 2018.

[53] J. Guo, K. Han, Y. Wang, H. Wu, X. Chen, C. Xu, and C. Xu, "Distilling object detectors via decoupled features," in *CVPR*, 2021.

[54] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *NIPS*, 2017.

[55] T. Wang, L. Yuan, X. Zhang, and J. Feng, "Distilling object detectors with fine-grained feature imitation," in *CVPR*, 2019.

[56] J. H. Cho, U. Mall, K. Bala, and B. Hariharan, "Picie: Unsupervised semantic segmentation using invariance and equivariance in clustering," in *CVPR*, 2021.

[57] X. Ji, J. F. Henriques, and A. Vedaldi, "Invariant information clustering for unsupervised image classification and segmentation," in *ICCV*, 2019.

[58] Y. Ouali, C. Hudelot, and M. Tami, "Autoregressive unsupervised image segmentation," in *ECCV*, 2020.

[59] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," in *NIPS*, 2016.

[60] D. Jha, M. A. Riegler, D. Johansen, P. Halvorsen, and H. D. Johansen, "Doubleu-net: A deep convolutional neural network for medical image segmentation," in *CBMS*, 2020.

[61] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[62] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: Redesigning skip connections to exploit multiscale features in image segmentation," *IEEE TMI*, 2019.

[63] D. Song, Y. Wang, H. Chen, C. Xu, C. Xu, and D. Tao, "Addersr: Towards energy efficient image super-resolution," in *CVPR*, 2021.

[64] X. Pan, X. Zhan, B. Dai, D. Lin, C. C. Loy, and P. Luo, "Exploiting deep generative prior for versatile image restoration and manipulation," *IEEE TPAMI*, 2021.

[65] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," *arXiv preprint arXiv:2102.12092*, 2021.

[66] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *COLING*, 2018.

[67] A. Adhikari, A. Ram, R. Tang, and J. Lin, "Docbert: Bert for document classification," *arXiv preprint arXiv:1904.08398*, 2019.

[68] W. J. Hutchins, *Machine translation: past, present, future*. Ellis Horwood Chichester, 1986.

[69] M. Simard, N. Ueffing, P. Isabelle, and R. Kuhn, "Rule-based translation with statistical phrase-based post-editing," in *WMT*, 2007.

[70] P. Koehn, *Statistical machine translation*. Cambridge University Press, 2009.

[71] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[72] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *EMNLP*, 2013.

[73] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NeurIPS*, 2014.

[74] J. Zhu, Y. Xia, L. Wu, D. He, T. Qin, W. Zhou, H. Li, and T. Liu, "Incorporating bert into neural machine translation," in *ICLR*, 2019.

[75] L. Hirschman and R. Gaizauskas, "Natural language question answering: the view from here," *Natural Language Engineering*, 2001.

[76] C. Toxtli, A. Monroy-Hernández, and J. Cranshaw, "Understanding chatbot-mediated task management," in *CHI*, 2018.

[77] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "Vqa: Visual question answering," in *ICCV*, 2015.

[78] E. Merdivan, D. Singh, S. Hanke, and A. Holzinger, "Dialogue systems for intelligent human computer interactions," *Electronic Notes in Theoretical Computer Science*, 2019.

[79] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, 2019.

[80] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," in *NeurIPS*, 2017.

[81] I. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio, "A hierarchical latent variable encoder-decoder model for generating dialogues," in *AAAI*, 2017.

[82] W. Yu, C. Zhu, Z. Li, Z. Hu, Q. Wang, H. Ji, and M. Jiang, "A survey of knowledge-enhanced text generation," *arXiv preprint arXiv:2010.04389*, 2020.

[83] S. Zhang, Z. Tan, J. Yu, Z. Zhao, K. Kuang, J. Liu, J. Zhou, H. Yang, and F. Wu, "Poet: Product-oriented video captioner for e-commerce." in *ACM MM*, 2020.

[84] R. Kosala and H. Blockeel, "Web mining research: A survey," *ACM Sigkdd Explorations Newsletter*, 2000.

[85] M. N. Sadiku, S. M. Musa, and O. D. Momoh, "Cloud computing: opportunities and challenges," *IEEE potentials*, 2014.

[86] A. Broder, "A taxonomy of web search," in *SIGIR*, 2002.

[87] F. Liu, C. Yu, W. Meng, and A. Chowdhury, "Effective keyword search in relational databases," in *SIGMOD*, 2006.

[88] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in *ECCV*, 2016.

[89] Z. Liu, C. Rodriguez-Opazo, D. Teney, and S. Gould, "Image retrieval on real-life images with pre-trained vision-and-language models," in *ICCV*, 2021.

[90] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *CVPR*, 2016.

[91] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, 1997.

[92] Z.-D. Zhao and M.-S. Shang, "User-based collaborative-filtering recommendation algorithms on hadoop," in *WKDD*, 2010.

[93] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *WWW*, 2001.

[94] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, 2009.

[95] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *DLRS Workshop*, 2016.

[96] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017.

[97] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: a factorization-machine based neural network for ctr prediction," in *IJCAI*, 2017.

[98] K. Cui, X. Chen, J. Yao, and Y. Zhang, "Variational collaborative learning for user probabilistic representation," in *AAAI Workshop*, 2018.

[99] J. Yao, Y. Zhang, I. Tsang, and J. Sun, "Discovering user interests from social images," in *MMM*, 2017.

[100] X. Chen, Y. Zhang, I. Tsang, Y. Pan, and J. Su, "Towards equivalent transformation of user preferences in cross domain recommendation," *arXiv preprint arXiv:2009.06884*, 2020.

[101] G. Shani, D. Heckerman, and R. I. Brafman, "An mdp-based recommender system," *JMLR*, 2005.

[102] D. Jannach and M. Ludewig, "When recurrent neural networks meet the neighborhood for session-based recommendation," in *RecSys*, 2017.
[103] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *ICDM*, 2018.
[104] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *KDD*, 2018.
[105] Q. Tan, J. Zhang, J. Yao, N. Liu, J. Zhou, H. Yang, and X. Hu, "Sparse-interest network for sequential recommendation," in *WSDM*, 2021.
[106] S. Zhang, D. Yao, Z. Zhao, T.-S. Chua, and F. Wu, "Causerec: Counterfactual user sequence synthesis for sequential recommendation," in *SIGIR*, 2021.
[107] Y. Lu, Y. Huang, S. Zhang, W. Han, H. Chen, Z. Zhao, and F. Wu, "Multi-trends enhanced dynamic micro-video recommendation." *CoRR*, 2021.
[108] Y. Pan, J. Yao, B. Han, K. Jia, Y. Zhang, and H. Yang, "Click-through rate prediction with auto-quantized contrastive learning," *arXiv preprint arXiv:2109.13921*, 2021.
[109] J. Huh and E. C. Malthouse, "Advancing computational advertising: Conceptualization of the field and future directions," *Journal of Advertising*, 2020.
[110] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web*, 2007.
[111] J. Turner, "The planning of guaranteed targeted display advertising," *Operations research*, 2012.
[112] E. M. Schwartz, E. T. Bradlow, and P. S. Fader, "Customer acquisition via display advertising using multi-armed bandit experiments," *Marketing Science*, 2017.
[113] Y. Chu, X. Wang, J. Ma, K. Jia, J. Zhou, and H. Yang, "Inductive granger causal modeling for multivariate time series," in *ICDM*, 2020.
[114] J. T. Yun, C. M. Segijn, S. Pearson, E. C. Malthouse, J. A. Konstan, and V. Shankar, "Challenges and future directions of computational advertising measurement systems," *Journal of Advertising*, 2020.
[115] M. B. Hoy, "Alexa, siri, cortana, and more: an introduction to voice assistants," *Medical reference services quarterly*, 2018.
[116] E. Pimenidis, N. Polatidis, and H. Mouratidis, "Mobile recommender systems: Identifying the major concepts," *Journal of Information Science*, 2019.
[117] J. Shuja, K. Bilal, W. Alasmary, H. Sinky, and E. Alanazi, "Applying machine learning techniques for caching in next-generation edge networks: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 181, p. 103005, 2021.
[118] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.
[119] J.-S. Park, H. Lee, D. Lee, J. Moon, S. Kwon, S. Ha, M. Kim, J. Park, J. Bang, S. Lim, and et al., "Samsung neural processing unit: An ai accelerator and sdk for flagship mobile ap." in *IEEE Hot Chips 33 Symposium, HCS 2021, Palo Alto, CA, USA, August 22-24, 2021*, 2021.
[120] K. J. Lee, "Chapter seven - architecture of neural processing unit for deep neural networks." *Adv. Comput.*, 2021.
[121] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018.
[122] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018.
[123] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *CVPR*, 2018.
[124] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.
[125] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Smash: one-shot model architecture search through hypernetworks," *arXiv preprint arXiv:1708.05344*, 2017.
[126] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast bayesian optimization of machine learning hyperparameters on large datasets," in *AISTATS*, 2017.
[127] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," *arXiv preprint arXiv:1804.09081*, 2018.
[128] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *AAAI*, 2018.
[129] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *ECCV*, 2018.
[130] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018.
[131] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Aging evolution for image classifier architecture search," in *AAAI*, 2019.
[132] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," *arXiv preprint arXiv:1902.08142*, 2019.
[133] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *ICML*, 2018.
[134] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.
[135] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
[136] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *CVPR*, 2018.
[137] J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," in *ICCV*, 2019.
[138] C. Yang, L. Xie, S. Qiao, and A. L. Yuille, "Training deep neural networks in generations: A more tolerant teacher educates better students," in *AAAI*, 2019.
[139] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *ICLR*, 2015.
[140] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *ICLR*, 2017.
[141] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *CVPR*, 2017.
[142] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *ICCV*, 2019.
[143] J. Kim, S. Park, and N. Kwak, "Paraphrasing complex network: Network compression via factor transfer," in *NeurIPS*, 2018.
[144] B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge transfer via distillation of activation boundaries formed by hidden neurons," in *AAAI*, 2019.
[145] P. Passban, Y. Wu, M. Rezagholizadeh, and Q. Liu, "Alp-kd: Attention-based layer projection for knowledge distillation," in *AAAI*, 2021.
[146] D. Chen, J.-P. Mei, Y. Zhang, C. Wang, Z. Wang, Y. Feng, and C. Chen, "Cross-layer distillation with semantic calibration," in *AAAI*, 2021.
[147] X. Wang, T. Fu, S. Liao, S. Wang, Z. Lei, and T. Mei, "Exclusivity-consistency regularized knowledge distillation for face recognition," in *ECCV*, 2020.
[148] L. Gan, Z. Teng, Y. Zhang, L. Zhu, F. Wu, and Y. Yang, "Semglove: Semantic co-occurrences for glove from bert," *arXiv preprint arXiv:2012.15197*, 2020.
[149] C. Zhang and Y. Peng, "Better and faster: knowledge transfer from multiple self-supervised learning tasks via graph distillation for video classification," in *IJCAI*, 2018.
[150] S. Lee and B. C. Song, "Graph-based knowledge distillation by multi-head attention network," in *BMVC*, 2019.
[151] R. Banner, I. Hubara, E. Hoffer, and D. Soudry, "Scalable methods for 8-bit training of neural networks," *arXiv preprint arXiv:1805.11046*, 2018.
[152] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *ICLR*, 2016.
[153] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh *et al.*, "Mixed precision training," *arXiv preprint arXiv:1710.03740*, 2017.
[154] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *ICML*, 2015.
[155] B. Ginsburg, S. Nikolaev, A. Kiswani, H. Wu, A. Gholaminejad, S. Kierat, M. Houston, and A. Fit-Florea, "Tensor processing using low precision format," Dec. 28 2017, uS Patent App. 15/624,577.
[156] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," *arXiv preprint arXiv:1412.7024*, 2014.
[157] B. Chmiel, L. Ben-Uri, M. Shkolnik, E. Hoffer, R. Banner, and D. Soudry, "Neural gradients are near-lognormal: improved quantized and sparse training," *arXiv preprint arXiv:2006.08173*, 2020.

[158] F. Faghri, I. Tabrizian, I. Markov, D. Alistarh, D. Roy, and A. Ramezani-Kebrya, "Adaptive gradient quantization for data-parallel sgd," *arXiv preprint arXiv:2010.12460*, 2020.
[159] Y. Li, X. Dong, and W. Wang, "Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks," *arXiv preprint arXiv:1909.13144*, 2019.
[160] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *NIPS*, 2015.
[161] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-oriented approximation of convolutional neural networks," *arXiv preprint arXiv:1604.03168*, 2016.
[162] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE transactions on neural networks and learning systems*, 2018.
[163] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *ECCV*, 2018.
[164] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio, "Neural networks with few multiplications," *arXiv preprint arXiv:1510.03009*, 2015.
[165] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, 2016.
[166] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.
[167] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Post-training 4-bit quantization of convolution networks for rapid-deployment," *arXiv preprint arXiv:1810.05723*, 2018.
[168] Y. Cai, Z. Yao, Z. Dong, A. Gholami, M. W. Mahoney, and K. Keutzer, "Zeroq: A novel zero shot quantization framework," in *CVPR*, 2020.
[169] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference." in *ICCV Workshops*, 2019.
[170] J. Fang, A. Shafiee, H. Abdel-Aziz, D. Thorsley, G. Georgiadis, and J. H. Hassoun, "Post-training piecewise linear quantization for deep neural networks," in *ECCV*, 2020.
[171] S. Garg, A. Jain, J. Lou, and M. Nahmias, "Confounding tradeoffs for neural network quantization," *arXiv preprint arXiv:2102.06366*, 2021.
[172] X. He and J. Cheng, "Learning compression from limited unlabeled data," in *ECCV*, 2018.
[173] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Improving post training neural quantization: Layer-wise calibration and integer programming," *arXiv preprint arXiv:2006.10518*, 2020.
[174] J. H. Lee, S. Ha, S. Choi, W.-J. Lee, and S. Lee, "Quantization for rapid deployment of deep neural networks," *arXiv preprint arXiv:1810.05488*, 2018.
[175] R. Zhao, Y. Hu, J. Dotzel, C. De Sa, and Z. Zhang, "Improving neural network quantization without retraining using outlier channel splitting," in *ICML*, 2019.
[176] A. Vysogorets and J. Kempe, "Connectivity matters: Neural network pruning through the lens of effective sparsity," *arXiv e-prints*, pp. arXiv–2107, 2021.
[177] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *NIPS*, 1990.
[178] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *ICNN*, 1993.
[179] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *NIPS*, 2015.
[180] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 4860–4874.
[181] S. Narang, E. Elsen, G. Diamos, and S. Sengupta, "Exploring sparsity in recurrent neural networks," *arXiv preprint arXiv:1704.05119*, 2017.
[182] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.
[183] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta, "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature communications*, vol. 9, no. 1, pp. 1–12, 2018.
[184] T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: Faster training without losing performance," *arXiv preprint arXiv:1907.04840*, 2019.
[185] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen, "Rigging the lottery: Making all tickets winners," in *International Conference on Machine Learning*. PMLR, 2020, pp. 2943–2952.
[186] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *International Conference on Learning Representations*, 2020.
[187] N. Lee, T. Ajanthan, and P. H. Torr, "Snip: Single-shot network pruning based on connection sensitivity," in *ICLR*, 2018.
[188] C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," in *International Conference on Learning Representations*, 2019.
[189] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," *Advances in Neural Information Processing Systems*, vol. 33, 2020.
[190] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *ICLR*, 2018.
[191] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *ICLR*, 2018.
[192] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, "Slimmable neural networks," in *ICLR*, 2018.
[193] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *ICPR*, 2016.
[194] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for efficient inference," in *ICML*, 2017.
[195] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *ECCV*, 2018.
[196] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *CVPR*, 2018.
[197] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *NIPS*, 2017.
[198] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM TIST*, 2019.
[199] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, 2020.
[200] K. C. Sim, P. Zadrazil, and F. Beaufays, "An investigation into on-device personalization of end-to-end automatic speech recognition models," in *ISCA*, 2019.
[201] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *arXiv preprint arXiv:2002.10619*, 2020.
[202] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage, "Federated evaluation of on-device personalization," *arXiv preprint arXiv:1910.10252*, 2019.
[203] Y. Jiang, J. Konecny, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv preprint arXiv:1909.12488*, 2019.
[204] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.
[205] T. Shen, J. Zhang, X. Jia, F. Zhang, G. Huang, P. Zhou, K. Kuang, F. Wu, and C. Wu, "Federated mutual learning," *arXiv preprint arXiv:2006.16765*, 2020.
[206] J. Hamer, M. Mohri, and A. T. Suresh, "Fedboost: A communication-efficient algorithm for federated learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3973–3983.
[207] C. Hou, K. K. Thekumparampil, G. Fanti, and S. Oh, "Reducing the communication cost of federated learning through multistage optimization," *arXiv preprint arXiv:2108.06869*, 2021.
[208] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.
[209] W. Y. B. Lim, J. S. Ng, Z. Xiong, J. Jin, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 536–550, 2021.
[210] W. Y. B. Lim, J. S. Ng, Z. Xiong, D. Niyato, C. Miao, and D. I. Kim, "Dynamic edge association and resource allocation in self-organizing hierarchical federated learning networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3640–3653, 2021.

[211] W. Y. B. Lim, S. Garg, Z. Xiong, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Uav-assisted communication efficient federated learning in the era of the artificial intelligence of things," *IEEE Network*, vol. 35, no. 5, pp. 188–195, 2021.

[212] W. Y. B. Lim, J. Huang, Z. Xiong, J. Kang, D. Niyato, X.-S. Hua, C. Leung, and C. Miao, "Towards federated learning in uav-enabled internet of vehicles: A multi-dimensional contract-matching approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5140–5154, 2021.

[213] Y. Yoon, D. Ban, S. Han, D. An, and E. Heo, "Device/cloud collaboration framework for intelligence applications," in *Internet of Things*, 2016.

[214] A. Banitalebi-Dehkordi, N. Vedula, J. Pei, F. Xia, L. Wang, and Y. Zhang, "Auto-split: A general framework of collaborative edge-cloud ai," in *KDD*, 2021.

[215] L. Hu, G. Sun, and Y. Ren, "Coedge: exploiting the edge-cloud collaboration for faster deep learning," *IEEE Access*, 2020.

[216] Y. Lu, Y. Shu, X. Tan, Y. Liu, M. Zhou, Q. Chen, and D. Pei, "Collaborative learning between cloud and end devices: an empirical study on location prediction," in *SEC*, 2019.

[217] J. Yao, F. Wang, K. Jia, B. Han, J. Zhou, and H. Yang, "Device-cloud collaborative learning for recommendation," in *KDD*, 2021.

[218] Z. Chen, J. Yao, F. Wang, K. Jia, B. Han, W. Zhang, and H. Yang, "Mc^2-sf: Slow-fast learning for mobile-cloud collaborative recommendation," *arXiv preprint arXiv:2109.12314*, 2021.

[219] L. Lyu, H. Yu, X. Ma, L. Sun, J. Zhao, Q. Yang, and P. S. Yu, "Privacy and robustness in federated learning: Attacks and defenses," *arXiv preprint arXiv:2012.06337*, 2020.

[220] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the Works of Leslie Lamport*, 2019.

[221] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," in *NeurIPS*, 2020.

[222] L. Zhu and S. Han, "Deep leakage from gradients," in *Federated learning*, 2020.

[223] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999.

[224] D. Demmler, T. Schneider, and M. Zohner, "Aby-a framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.

[225] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, 2014.

[226] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv preprint arXiv:1711.10677*, 2017.

[227] A. C. Yao, "Protocols for secure computations," in *FOCS*, 1982.

[228] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "Ldp-fed: Federated learning with local differential privacy," in *EdgeSys Workshop*, 2020.

[229] S. Dey, J. Mondal, and A. Mukherjee, "Offloaded execution of deep learning inference at edge: Challenges and insights," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2019, pp. 855–861.

[230] Z. Xu, L. Zhao, W. Liang, O. F. Rana, P. Zhou, Q. Xia, W. Xu, and G. Wu, "Energy-aware inference offloading for dnn-driven applications in mobile edge clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 799–814, 2020.

[231] R. G. Pacheco, R. S. Couto, and O. Simeone, "Calibration-aided edge inference offloading via adaptive model partitioning of deep neural networks," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[232] R. G. Pacheco, F. D. Oliveira, and R. S. Couto, "Early-exit deep neural networks for distorted images: Providing an efficient edge offloading," *arXiv preprint arXiv:2108.09343*, 2021.

[233] S. Zhou, W. Jadoon, and J. Shuja, "Machine learning-based offloading strategy for lightweight user mobile edge computing tasks," *Complexity*, vol. 2021, 2021.

[234] J. Shuja, S. Mustafa, R. W. Ahmad, S. A. Madani, A. Gani, and M. K. Khan, "Analysis of vector code offloading framework in heterogeneous cloud and edge architectures," *IEEE Access*, vol. 5, pp. 24 542–24 554, 2017.

[235] A. Asheralieva, D. Niyato, and Z. Xiong, "Auction-and-learning based lagrange coded computing model for privacy-preserving, secure, and resilient mobile edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[236] J. S. Ng, W. Y. B. Lim, Z. Xiong, D. Niyato, C. Leung, and C. Miao, "A double auction mechanism for resource allocation in coded vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1832–1845, 2022.

[237] H. Li, C. Hu, J. Jiang, Z. Wang, Y. Wen, and W. Zhu, "Jalad: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution," in *ICPADS*, 2018.

[238] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms," in *AVSS*, 2018.

[239] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in uav-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2022.

[240] D. Kahneman, *Thinking, fast and slow*. Macmillan, 2011.

[241] K. Madan, R. N. Ke, A. Goyal, B. B. Schölkopf, and Y. Bengio, "Fast and slow learning of recurrent independent mechanisms," in *ICLR*, 2021.

[242] F. Zhang, X. Qi, R. Yang, V. Prisacariu, B. W. Wah, and P. H. S. Torr, "Domain-invariant stereo matching networks," in *ECCV*, 2020.

[243] Y.-R. Yeh, C.-H. Huang, and Y.-C. F. Wang, "Heterogeneous domain adaptation and classification by exploiting the correlation subspace," *IEEE TIP*, 2014.

[244] H. Wang, X. Wu, and Y. Jia, "Heterogeneous domain adaptation method for video annotation," *IET-CVI*, 2017.

[245] C.-X. Ren, J. Feng, D.-Q. Dai, and S. Yan, "Heterogeneous domain adaptation via covariance structured feature translators," *IEEE Trans. Cybern.*, 2021.

[246] H. Wu, H. Zhu, Y. Yan, J. Wu, Y. Zhang, and M. K. Ng, "Heterogeneous domain adaptation by information capturing and distribution matching," *IEEE TIP*, 2021.

[247] Y.-H. H. Tsai, Y.-R. Yeh, and Y.-C. F. Wang, "Learning cross-domain landmarks for heterogeneous domain adaptation," in *CVPR*, 2016.

[248] A. Samat, C. Persello, P. Gamba, S. Liu, J. Abuduwaili, and E. Li, "Supervised and semi-supervised multi-view canonical correlation analysis ensemble for heterogeneous domain adaptation in remote sensing image classification," *Remote. Sens.*, 2017.

[249] B. Li, Y. Wang, S. Zhang, D. Li, K. Keutzer, T. Darrell, and H. Zhao, "Learning invariant representations and risks for semi-supervised domain adaptation," in *CVPR*, 2021.

[250] X. Wang, Y. Ma, Y. Cheng, L. Zou, and J. J. P. C. Rodrigues, "Heterogeneous domain adaptation network based on autoencoder," *JPDC*, 2018.

[251] F. Liu, G. Zhang, H. Lu, and J. Lu, "Heterogeneous unsupervised cross-domain transfer learning," *arXiv preprint arXiv:1701.02511*, 2017.

[252] J. T. Zhou, I. W. Tsang, S. J. Pan, and M. Tan, "Heterogeneous domain adaptation for multiple classes," in *AISTATS*, 2014.

[253] B. Kulis, K. Saenko, and T. Darrell, "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms," in *CVPR*, 2011.

[254] K. D. Feuz and D. J. Cook, "Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (fsr)," *ACM Trans. Intell. Syst. Technol.*, 2015.

[255] M. Xiao and Y. Guo, "Feature space independent semi-supervised domain adaptation via kernel matching," *IEEE TPAMI*, 2015.

[256] H. Wu, Q. Wu, and M. K. Ng, "Knowledge preserving and distribution alignment for heterogeneous domain adaptation," *ACM TOIS*, 2022.

[257] B. Rosenfeld, B. Rajendran, and O. Simeone, "Fast on-device adaptation for spiking neural networks via online-within-online meta-learning," *arXiv preprint arXiv:2103.03901*, 2021.

[258] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.

[259] L. Huang, L. Zhang, S. Yang, L. P. Qian, and Y. Wu, "Meta-learning based dynamic computation task offloading for mobile edge computing networks," *IEEE Commun. Lett.*, 2021.

[260] W. Zhou, C. Xu, and J. J. McAuley, "Meta learning for knowledge distillation," *arXiv preprint arXiv:2106.04570*, 2021.

[261] J. Ye, S. Zhang, and J. Wang, "Hybrid network compression via meta-learning," in *MM*, 2021.

[262] H. Pan, C. Wang, M. Qiu, Y. Zhang, Y. Li, and J. Huang, "Meta-kd: A meta knowledge distillation framework for language model compression across domains," in *ACL/IJCNLP*, 2021.

[263] M. Zhang, D. Wang, and S. Gai, "Knowledge distillation for model-agnostic meta-learning," in *ECAI*, 2020.

[264] J. Pearl, *Causality: Models, Reasoning, and Inference*, 2009.

[265] K. Kuang, L. Li, Z. Geng, L. Xu, K. Zhang, B. Liao, H. Huang, P. Ding, W. Miao, and Z. Jiang, "Causal inference," *Engineering*, 2020.

[266] G. Rotman, A. Feder, and R. Reichart, "Model compression for domain adaptation through causal effect estimation," *arXiv preprint arXiv:2101.07086*, 2021.

[267] T. Teshima, I. Sato, and M. Sugiyama, "Few-shot domain adaptation by causal mechanism transfer," in *ICML*, 2020.

[268] M. Yang, Y. Shen, X. Chen, and C. Li, "Multi-source domain adaptation for sentiment classification with granger causal inference," in *SIGIR*, 2020.

[269] Y. Chen and P. Bühlmann, "Domain adaptation under structural causal models," *arXiv preprint arXiv:2010.15764*, 2020.

[270] Z. Yue, Q. Sun, X.-S. Hua, and H. Zhang, "Transporting causal mechanisms for unsupervised domain adaptation," in *ICCV*, 2021.

[271] K. Kuang, P. Cui, S. Athey, R. Xiong, and B. Li, "Stable prediction across unknown environments," in *KDD*, 2018.

[272] J. Yuan, X. Ma, K. Kuang, R. Xiong, M. Gong, and L. Lin, "Learning domain-invariant relationship with instrumental variable for domain generalization," *arXiv preprint arXiv:2110.01438*, 2021.

[273] S. Yang, K. Yu, F. Cao, L. Liu, H. Wang, and J. Li, "Learning causal representations for robust domain adaptation," *IEEE TKDE*, 2021.

[274] S. Zhang, T. Jiang, T. Wang, K. Kuang, Z. Zhao, J. Zhu, J. Yu, H. Yang, and F. Wu, "Devlbert: Learning deconfounded visio-linguistic representations," in *MM*, 2020, pp. 4373–4382.

[275] K. Yu, X. Guo, L. Liu, J. Li, H. Wang, Z. Ling, and X. Wu, "Causality-based feature selection: Methods and evaluations," *ACM Comput. Surv.*, 2020.

[276] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. S. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. D. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. S. Krass, R. Krishna, R. Kuditipudi, and et al., "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[277] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *NeurIPS*, 2020.

[278] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[279] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "Gshard: Scaling giant models with conditional computation and automatic sharding," in *ICLR*, 2021.

[280] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *arXiv preprint arXiv:2101.03961*, 2021.

[281] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *ICLR*, 2020.

[282] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[283] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "TinyBERT: Distilling BERT for natural language understanding," in *EMNLP*, 2020.

[284] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "MobileBERT: a compact task-agnostic BERT for resource-limited devices," in *ACL*, 2020.

[285] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," in *NeurIPS*, 2020.

[286] W. Zhou, C. Xu, T. Ge, J. J. McAuley, K. Xu, and F. Wei, "Bert loses patience: Fast and robust inference with early exit," in *NeurIPS*, 2020.

[287] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," *arXiv preprint arXiv:2109.01652*, 2021.

[288] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," in *ACL*, 2021.

[289] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE TNNLS*, 2020.

[290] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph r-cnn for scene graph generation," in *ECCV*, 2018.

[291] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *CVPR*, 2018.

[292] D. Marcheggiani, J. Bastings, and I. Titov, "Exploiting semantics in neural machine translation with graph convolutional networks," in *NAACL*, 2018.

[293] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," in *ACL*, 2018.

[294] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *AAAI*, 2018.

[295] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *ICLR*, 2017.

[296] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *AAAI*, 2019.

[297] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *NIPS*, 2015.

[298] M. Jiang, T. Jung, R. Karl, and T. Zhao, "Federated dynamic gnn with secure aggregation," *arXiv preprint arXiv:2009.07351*, 2020.

[299] J. Zhou, C. Chen, L. Zheng, X. Zheng, B. Wu, Z. Liu, and L. Wang, "Privacy-preserving graph neural network for node classification," *arXiv e-prints*, 2020.

[300] L. Zheng, J. Zhou, C. Chen, B. Wu, L. Wang, and B. Zhang, "Asfgnn: Automated separated-federated graph neural network," *Peer-to-Peer Networking and Applications*, 2021.

[301] B. Wang, A. Li, H. Li, and Y. Chen, "Graphfl: A federated learning framework for semi-supervised node classification on graphs," *arXiv preprint arXiv:2012.04187*, 2020.

[302] C. Wu, F. Wu, Y. Cao, Y. Huang, and X. Xie, "Fedgnn: Federated graph neural network for privacy-preserving recommendation," *arXiv preprint arXiv:2102.04925*, 2021.

[303] C. He, K. Balasubramanian, E. Ceyani, Y. Rong, P. Zhao, J. Huang, M. Annavaram, and S. Avestimehr, "Fedgraphnn: A federated learning system and benchmark for graph neural networks," *arXiv preprint arXiv:2104.07145*, 2021.

[304] C. Wang, B. Chen, G. Li, and H. Wang, "Fl-agcns: Federated learning framework for automatic graph convolutional network search," *arXiv preprint arXiv:2104.04141*, 2021.

[305] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodolà, and B. Caputo, "Cluster-driven graph federated learning over multiple domains," in *CVPR*, 2021.

[306] C. Chen, W. Hu, Z. Xu, and Z. Zheng, "Fedgl: Federated graph learning framework with global self-supervision," *arXiv preprint arXiv:2105.03170*, 2021.

[307] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr, "Spreadgnn: Serverless multi-task federated learning for graph neural networks," *arXiv preprint arXiv:2106.02743*, 2021.

[308] C. Meng, S. Rambhatla, and Y. Liu, "Cross-node federated graph neural network for spatio-temporal data modeling," in *KDD*, 2021.

[309] X. Ni, X. Xu, L. Lyu, C. Meng, and W. Wang, "A vertical federated learning framework for graph convolutional network," *arXiv preprint arXiv:2106.11593*, 2021.

[310] H. Xie, J. Ma, L. Xiong, and C. Yang, "Federated graph classification over non-iid graphs," *arXiv preprint arXiv:2106.13423*, 2021.

[311] K. Zhang, C. Yang, X. Li, L. Sun, and S. M. Yiu, "Subgraph federated learning with missing neighbor generation," *arXiv preprint arXiv:2106.13430*, 2021.

[312] B. Feng, Y. Wang, X. Li, S. Yang, X. Peng, and Y. Ding, "Sgquant: Squeezing the last bit on graph neural networks with specialized quantization," in *ICTAI*, 2020.

[313] S. A. Tailor, J. Fernandez-Marques, and N. D. Lane, "Degree-quant: Quantization-aware training for graph neural networks," in *ICLR*, 2021.

[314] H. Wang, D. Lian, Y. Zhang, L. Qin, X. He, Y. Lin, and X. Lin, "Binarized graph neural network," *World Wide Web*, 2021.

[315] M. Bahri, G. Bahl, and S. Zafeiriou, "Binary graph neural networks," in *CVPR*, 2021.

[316] J. Wang, Y. Wang, Z. Yang, L. Yang, and Y. Guo, "Bi-gcn: Binary graph convolutional network," in *CVPR*, 2021.

[317] D. Mandal, S. Medya, B. Uzzi, and C. Aggarwal, "Meta-learning with graph neural networks: Methods and applications," *arXiv preprint arXiv:2103.00137*, 2021.

[318] K. Huang and M. Zitnik, "Graph meta learning via local subgraphs," *NeurIPS*, 2020.

[319] N. Wang, M. Luo, K. Ding, L. Zhang, J. Li, and Q. Zheng, "Graph few-shot learning with attribute matching," in *CIKM*, 2020.

[320] Z. Wen, Y. Fang, and Z. Liu, "Meta-inductive node classification across graphs," in *SIGIR*, 2021.

[321] N. Ma, J. Bu, J. Yang, Z. Zhang, C. Yao, Z. Yu, S. Zhou, and X. Yan, "Adaptive-step graph meta-learner for few-shot graph classification," in *CIKM*, 2020.

[322] S. Jiang, F. Feng, W. Chen, X. Li, and X. He, "Structure-enhanced meta-learning for few-shot graph classification," *arXiv preprint arXiv:2103.03547*, 2021.

[323] D. Buffelli and F. Vandin, "A meta-learning approach for graph representation learning in multi-task settings," *arXiv preprint arXiv:2012.06755*, 2020.

[324] Y. Li, "Deep reinforcement learning: An overview," *ArXiv*, vol. abs/1701.07274, 2018.

[325] C. Mouradian and S. Y. R. H. Glitho, "Robots as-a-service in cloud computing: Search and rescue in large-scale disasters case study," *IEEE CCNC*, 2018.

[326] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 12:1–12:19, 2019. [Online]. Available: https://doi.org/10.1145/3298981

[327] H. Zhuo, W. Feng, Q. Xu, Q. Yang, and Y. Lin, "Federated reinforcement learning," *ArXiv*, 2019.

[328] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *INFOCOM*, 2020.

[329] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5g ultradense network," *IEEE Internet of Things Journal*, 2021.

[330] F. X. Fan, Y. Ma, Z. Dai, W. Jing, C. Fan, and B. K. H. Low, "Fault-tolerant federated reinforcement learning with theoretical guarantee," in *NIPS*, 2021.

[331] B. Liu, L. Wang, and M. Liu, "Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems," *IEEE Robotics and Automation Letters*, 2019.

[332] J. Qi, Q. Zhou, L. Lei, and K. Zheng, "Federated reinforcement learning: Techniques, applications, and open challenges," *arXiv preprint arXiv:2108.11887*, 2021.

[333] Y. Hu, Y. Hua, W. Liu, and J. Zhu, "Reward shaping based federated reinforcement learning," *IEEE Access*, 2021.

[334] A. Anwar and A. Raychowdhury, "Multi-task federated reinforcement learning with adversaries," *arXiv preprint arXiv:2103.06473*, 2021.

[335] J. Wu, G. Zhang, J. Nie, Y. Peng, and Y. Zhang, "Deep reinforcement learning for scheduling in an edge computing-based industrial internet of things," *Wireless Communications and Mobile Computing*, 2021.

[336] S. Sheng, P. Chen, Z. Chen, L. Wu, and Y. Yao, "Deep reinforcement learning-based task scheduling in iot edge computing," *Sensors*, 2021.

[337] T. Zheng, J. Wan, J. Zhang, and J. Congfeng, "Reinforcement learning-based workload scheduling for edge computing," 2021.

[338] R. Zhao, X. Wang, J. Xia, and L. Fan, "Deep reinforcement learning based mobile edge computing for intelligent internet of things," *Phys. Commun.*, 2020.

[339] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," *EURASIP J. Wirel. Commun. Netw.*, 2018.

[340] Y. Dai, D. Xu, K. Zhang, Y. Lu, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for edge computing and resource allocation in 5g beyond," in *ICCT*, 2019.

[341] G. Qu, H. Wu, R. Li, and P. Jiao, "Dmro:a deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE TNSM*, 2020.

[342] H. Hao, C. Xu, L. Zhong, and G.-M. Muntean, "A multi-update deep reinforcement learning algorithm for edge computing service offloading," in *MM*, 2020.

[343] W. Zhan, C. Luo, J. Wang, C. Wang, G. Min, H. Duan, and Q. Zhu, "Deep reinforcement learning-based offloading scheduling for vehicular edge computing," *IEEE Internet of Things*, 2020.

[344] X. Wang, W. Wang, and Z. Jin, "Context-aware reinforcement learning-based mobile cloud computing for telemonitoring," in *BHI*, 2018.

[345] S. W. Park, A. Boukerche, and S. Guan, "A novel deep reinforcement learning based service migration model for mobile edge computing," in *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2020, pp. 1–8.

[346] Y. Li, H. Chen, Z. Fu, Y. Ge, and Y. Zhang, "User-oriented fairness in recommendation," in *WWW*, 2021.

[347] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, "Bias and debias in recommender system: A survey and future directions," *arXiv preprint arXiv:2010.03240*, 2020.

[348] H. Shin, S. Kim, J. Shin, and X. Xiao, "Privacy enhanced matrix factorization for recommendation with local differential privacy," *IEEE TKDE*, 2018.

[349] K. Muhammad, Q. Wang, D. O'Reilly-Morgan, E. Tragos, B. Smyth, N. Hurley, J. Geraci, and A. Lawlor, "Fedfast: Going beyond average for faster training of federated recommender systems," in *KDD*, 2020.

[350] A. Freno, M. Saveski, R. Jenatton, and C. Archambeau, "One-pass ranking models for low-latency product recommendations," in *KDD*, 2015.

[351] M. Khatun, M. Glaß, and R. Jung, "An approach of scenario-based threat analysis and risk assessment over-the-air updates for an autonomous vehicle," in *ICARA*, 2021.

[352] B. Vaidya, P. P. Kaur, and H. T. Mouftah, "Provisioning road weather management using edge cloud and connected and autonomous vehicles," in *IWCMC*, 2021.

[353] Y. Deng, T. Zhang, G. Lou, X. Zheng, J. Jin, and Q.-L. Han, "Deep learning-based autonomous driving systems: A survey of attacks and defenses," *IEEE Trans. Ind. Informatics*, 2021.

[354] D. V. Nguyen, H. T. T. Tran, and T. C. Thang, "A delay-aware adaptation framework for cloud gaming under the computation constraint of user devices," in *MMM*, 2020.

[355] M. Basiri and A. Rasoolzadegan, "Delay-aware resource provisioning for cost-efficient cloud gaming," *IEEE TCSVT*, 2018.

[356] S. Kassir, G. d. Veciana, N. Wang, X. Wang, and P. Palacharla, "Joint update rate adaptation in multiplayer cloud-edge gaming services: Spatial geometry and performance tradeoffs," in *MobiHoc*, 2021.

[357] J. D. N. Dionisio, I. Burns, William G., and R. Gilbert, "3d virtual worlds and the metaverse: Current status and future possibilities." *ACM Comput. Surv.*, 2013.

[358] L.-H. Lee, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo, and P. Hui, "All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda." *CoRR*, 2021.

[359] N. Waheed, X. He, M. Ikram, M. Usman, S. S. Hashmi, and M. Usman, "Security and privacy in iot using machine learning and blockchain: Threats and countermeasures," *ACM Comput. Surv.*, 2021.

[360] M. R. Shahid, "Deep learning for internet of things (iot) network security. (apprentissage profond (deep learning) pour la sécurité des réseaux d'objets connectés (iot))," Ph.D. dissertation, 2021.

[361] Z. Xiong, J. Kang, D. Niyato, P. Wang, and H. V. Poor, "Cloud/edge computing service management in blockchain networks: Multi-leader multi-follower game-based admm for pricing," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 356–367, 2020.

[362] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/fog computing resource management and pricing for blockchain networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4585–4600, 2019.