Digital Communications and Networks 8 (2022) 1–17

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

A survey on deploying mobile deep learning applications: A systemic and technical perspective

Yingchun Wang [a], Jingyi Wang [a], Weizhan Zhang [a,*], Yufeng Zhan [b], Song Guo [b], Qinghua Zheng [a], Xuanyu Wang [a]

[a] MOEKLINNS Lab, School of Computer Science and Technology, Xi'an Jiaotong University, Shaanxi, 710049, China
[b] Department of Computing, The Hong Kong Polytechnic University, HK, 999077, China

ARTICLE INFO

Keywords:
Deep learning
Mobile computing
Distributed offloading
Distributed caching

ABSTRACT

With the rapid development of mobile devices and deep learning, mobile smart applications using deep learning technology have sprung up. It satisfies multiple needs of users, network operators and service providers, and rapidly becomes a main research focus. In recent years, deep learning has achieved tremendous success in image processing, natural language processing, language analysis and other research fields. Despite the task performance has been greatly improved, the resources required to run these models have increased significantly. This poses a major challenge for deploying such applications on resource-restricted mobile devices. Mobile intelligence needs faster mobile processors, more storage space, smaller but more accurate models, and even the assistance of other network nodes. To help the readers establish a global concept of the entire research direction concisely, we classify the latest works in this field into two categories, which are local optimization on mobile devices and distributed optimization based on the computational position of machine learning tasks. We also list a few typical scenarios to make readers realize the importance and indispensability of mobile deep learning applications. Finally, we conjecture what the future may hold for deploying deep learning applications on mobile devices research, which may help to stimulate new ideas.

1. Introduction

The development of smart phones, laptops and other new mobile devices has further promoted the development of AI applications on mobile devices. In this paper, we define a deep model that has been trained and applied to a specific service and is designed to run on mobile devices as Mobile Deep Learning Applications(MDLA). Its training may be cloud-based, or it may be based on edge devices using federated learning technology, which is not our focus. The focus of our investigation is the reasoning process of these mobile deep learning models. Cameras, microphones, and sensors can obtain various types of information like video, audio, and acceleration from the real world. These kinds of data are then provided to MDLAs. Based on this, MDLAs have developed rapidly and attracted widespread attention due to their tangible benefits for users from all sides. For example, MDLAs benefit users by performing malicious software detection [1], app recommendation [2], user verification [3,4], mobile visual tasks [5,6], mobile web browsing optimization [7], human activity monitoring [8], medical health monitoring [9,10] and other smart fields [11–17].

For network operators and third-party service providers, the deployment of MDLA can support mobile crowdsourcing scenarios [18–21], distributed machine learning [22–24], federated learning [25], multiple smart IoT applications [26], and other fantastic services using mobile big data [27,28], etc. The intelligence in mobile applications is changing the way people live, work and interact with the world.

Deep learning is the main method of MDLAs. Even though these smart mobile vision applications are wonderful, they require a lot of storage, calculations, and high consumption of power and network bandwidth, and users may be reluctant to download them to their mobile devices. These requirements, in contradiction with the limited resources of mobile devices, become the main bottleneck for the development of MDLAs. For example, CNN, serving as the main method in mobile vision tasks, is executed for each input as a cascade of layers, mainly including convolution layers, pooling layers and FC, and it produces intermediate results called feature maps, and outputs inference results. Such CNN executions are known for their high time and space complexity. A typical CNN

* Corresponding author.
E-mail address: zhangwzh@xjtu.edu.cn (W. Zhang).

https://doi.org/10.1016/j.dcan.2021.06.001
Received 4 June 2020; Received in revised form 6 May 2021; Accepted 1 June 2021
Available online 16 June 2021

model, such as AlexNet, occupying more than 200 M of memory, has 60 million parameters and needs 720 million FLOPs. The other, VGG-16, occupying more than 500 MB of memory, has 138 million parameters and needs 15 300 million FLOPs. Even though these smart mobile vision applications are wonderful, they require a lot of storage, calculations, and high consumption of power and network bandwidth, yet users may be reluctant to download them to their mobile devices.

It is very important to study the deployment of MDLAs on mobile devices, which makes it possible to migrate a large number of centralized applications to the mobile end. Some of them have changed our daily lives. Users may need to manually record their meal information in the past, but now it can be achieved by just using a smart spoon. Besides, there are many other important public scenes. For example, by carefully designing the offloading strategy, Lu et al. make it possible to deploy CNN on the user's mobile phone to detect important targets like criminals by crowd sensing. Without MDLAs, the intelligence of the machine will stay in the centralized cloud and will never appear in front of people at the edge. So our work is dedicated to investigating some classic and state-of-the-art research on deploying MDLA.

Recently, two methods have been explored to solve this problem. The first method solves this problem from the perspective of the software and hardware of local mobile devices, which means that the goal is to run MDLAs locally on mobile devices without the help of a third party. The key method is to reduce the resource requirements for running a deep learning model or optimize mobile device hardware that is more suitable for MDLAs. There are some popular solutions. One approach is to compress the deep learning model. Even though this might influence its accuracy, it decreases the demand for computation and storage resources. An essential problem is how to balance the two counterparts [29–35]. Another possible solution is to reduce computational needs by reusing intermediate computing results [6,29,36,37] or maximizing the rate of utilizing device resources through precise dispatching among multiple deep learning tasks [5,38,39]. Moreover, much work has been done to develop deep learning frameworks suitable for mobile devices [40–46]. Finally, improving the hardware of mobile devices to support the operation of MDLAs can be taken into consideration.

Another research direction is to gain support from background servers with sufficient resources. Here, "sufficient resources" means that the current resources of background servers are sufficient for the running of tasks to be offloaded and that the servers are not limited to cloud servers. The target offloading servers can be mobile edge computing servers, peer devices, or any places that are both resource-sufficient and network reachable. First, in 2006, Amazon launched the Elastic Compute Cloud (EC2) [47], and the concept of cloud computing was proposed. Offloading to cloud servers with abundant computation and storage resources can remarkably reduce computing delay. However, this approach might introduce a major network delay due to the unstable and restricted bandwidth of the radio channel. Hence, how to reduce transfer latency has become the focus of mobile offloading systems based on clouds [48–50]. Second, in 2014, ETSI and IBM jointly established the Mobile Edge Computing(MEC) standardization group and formally proposed the concept of standardized MEC [51]. MEC offers IT and cloud computing in a Radio Access Network (RAN); its location is closer to mobile users, which further reduces transfer latency. However, due to the limited resources compared with cloud resources, problems such as multi-user resource allocation [52–54], service placement location [55], and offloading task allocation on multiple servers [56] have appeared. Third, the emergence of ad hoc cloudlets and D2D communication in recent years enables mobile devices to transfer computing to other nearby computing devices like smart phones, home-use computers, etc. [57]. Even though such a strategy minimizes the distance between task-initiated devices and their computing location, considering the complicated mobility of users and the resource availability of the cloudlet, the offloading strategy needs to be delicately designed within its limited coverage area. In addition, the caching of services and data on background servers and communication optimization between mobile devices and backgrounds are also key research areas, but these are not the focus of this paper.

Fig. 1 shows an overview of deploying MDLAs. Mobile users can execute local computation or choose distributed deployment (by offloading). The up arrow indicates the offloading process. Computing tasks from mobile devices are firstly offloaded to the edge server. If the edge server cannot satisfy its resource needs, it can be further offloaded to the remote cloud with sufficient resources. We can also use D2D communication to transfer some relatively light computing to devices of ad-hoc cloudlets due to the closer distance.

Some investigations have been carried out in related fields. Mao's work [58] mainly discusses mobile edge computing from the perspectives of communication technology and computing resources, and studies the offloading of mobile computing to edge servers through computing architecture, communication models and resource management. Another work that also focuses on MEC [59] introduces some mobile edge computing cases and service environments, followed by a detailed illustration of the standardized MEC infrastructure. Finally, this work discusses three key areas of MEC computation offloading: offloading strategy, edge server resource allocation and user mobility management. Dao's work [60] discusses how to deploy multimedia applications using deep learning on mobile devices. This paper focuses on the local computation of deep multimedia applications on mobile devices and discusses it from two aspects of software and hardware. Kumar's work [61] focuses on the distributed deployment of mobile computing tasks, surveying it from the perspective of computation offloading. However, in these works, we can learn the running of MDLA only based on certain aspects: offloading its computation to certain places such as edge servers, only learning distributed deployment using offloading techniques and only discussing a special kind of MDLA such as deep multimedia applications. None of these works studied MDLAs in a comprehensive way.

This paper studies MDLAs from perspectives of systematization and networking, and conducts a comprehensive survey on the challenges and corresponding state-of-the-art solutions from two directions: local optimized running and distributed deployment. The rest of this paper is organized as follows. Section 2 introduces the deployment of MDLAs on mobile devices; Section 3 illustrates the distributed deployment of MDLA; Section 4 classifies MDLAs according to different beneficiaries; Section 5 lists future possible research directions of related fields; Section 6 summarizes this article.

2. Deployment on mobile devices

In this section, we discuss how to run MDLAs locally on mobile devices. Its main idea is to reduce the resource requirements for running deep learning tasks or to design deep learning frameworks suitable for mobile devices to optimize an MDLA's computation. To reduce resource usage, we can make the following considerations: firstly, the natural idea

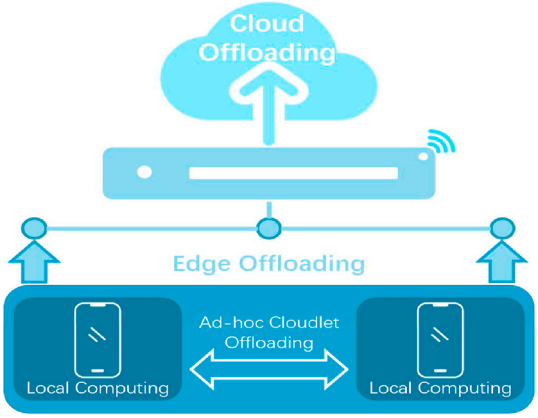

Fig. 1. Overview of the deployment of MDLAs.

is to reduce the amount of calculation and storage space required by the model itself; second, for different input data, we can consider the reuse of some calculation results to reduce the computing resource requirements; what's more, there are usually multiple deep learning tasks running on a single mobile device. By well-dispatching resources between multiple deep tasks, the purpose of "resource-saving" can also be achieved. The details are as follows.

2.1. Reducing the deep learning model's complexity

Deep learning tasks often involve large models and massive computation. Hence, when running MDLAs locally, problems such as insufficient memory or computational resources are quite obvious. A direct solution is to compress deep learning models so that they can be executed completely on mobile devices. Currently, this compression method is widely used in deep neural networks. Although the types of deep neural networks are different, fundamentally, they comprise a general feature extraction part and a decision part for certain tasks. The bottom feature extraction layers dominate computational costs due to their relatively heavy computation, while the decision layer dominates storage costs due to the large number of parameters that strongly influence the model size. There are many kinds of research about model compressing in recent years, and this paper summarizes them based on space-memory and time-computation.

2.1.1. Reducing model spatial complexity

The spatial complexity of deep neural networks is determined by the number and size of parameters in the deep model. By reducing the spatial complexity of the deep model, its memory can be greatly reduced. Based on this, we can classify related works into two categories: cutting down the number of model parameters, including pruning and sharing, and reducing its size like weight quantization. Specific research contents are as follows.

2.1.1.1. Pruning. Pruning aims to reduce the number of model parameters. The basic idea is to select and delete some trivial parameters that have little influence on the model's accuracy, and then retrain the model to recover the model performance.

Nonstructural pruning removes trivial neurons to refine the model size, including the parameters of layer size and the network depth. For instance, in Han's work, inessential connections in a trained network will be pruned to reduce parameters, thus reducing memory and CPU consumption [30].

Structural pruning exploits the structural sparsity of the model at different scales, including filter sparsity, kernel sparsity, and feature mapping sparsity, which saves computation resources for embedded devices, parallel computing and distributed systems. In Anwar's work [31], for example, they propose a particle filtering approach to determine the importance of different network connections. The weight of importance of each particle is assigned by computing the misclassification rate of the corresponding connectivity pattern. The pruned network is retrained to compensate for the loss caused by pruning. In Fang's work NESTDNN [5], the offline phase includes three subphases: model pruning, model recovery, and model analysis. In the model pruning phase, NestDNN adopts the state-of-the-art Triplet Response Residual (TRR) approach to sort the filters in the given deep learning model according to their relative importance and then prunes the filters iteratively. In each iteration, the approach prunes relatively unimportant filters and retrains the pruned model to compensate for the loss of accuracy. The iteration ends when the pruned model fails to reach the minimum accuracy set by the users.

Middle hidden-layer pruning directly deletes some layers in networks. Rueda et al. [32] propose a method to maximize units to combine neurons into more complex convex functions and select neurons based on the local relevance of each neuron's activation on the training set. Li et al. [62] divide the network layers into weight layers, such as convolutional or fully connected layers, and non-weight layers, such as pooling or activation layers. The non-weight layer incurs less theoretical computation but a longer computation time due to memory data access time and for other reasons. The authors combine a non-weight layer with a weight layer and eliminate independent non-weight layers, which significantly reduces the inference time.

2.1.1.2. Parameter sharing. There are weighted edges between layers, and the number of weight parameters will increase with a rise in the number of nodes. Therefore, to reduce parameters, approaches are designed to share the weights of certain edges. A simple example is that if every layer has 1000 nodes, then we need to store 1000×1000 wt parameters. However, if we cluster edges with approximate weights, the model size will be sharply reduced. Similarly, if we have 1000 cluster centers, we only have to store 1000 parameters in this way.

For example, the main idea of Han's work [63] is to compute the multiple clustering centers of weights using *k*-means clustering. This approach clusters the weights to the nearest centroids and compensates the weight through fine-tuning training. Chen hashes parameters into corresponding hash buckets, and parameters in the same bucket share a single value [33].

2.1.1.3. Network quantization. Network quantization compresses initial networks by reducing the number of bits for each weight parameter. Quantization uses low precision data to represent the original high precision data. The traditional technique is to convert 32-bit floating-point to 16-bit floating-point, 16-bit fixed-point, 8-bit fixed-point, etc. There are some related technologies like Binarized Neural Networks, Ternary Weight Networks, XNOR Networks. In Li's work [34], he proposes a higher-order binarization scheme and executes binarization both on the input layer and the weight layer of the CNN. Specifically, this method recursively performs residual quantization to improve the effect of binary approximation. Yin et al. accelerate the operation of DNN on low-power computing hardware by understanding Straight-Through Estimator in training activation quantized neural nets [64].

2.1.1.4. Direct design of small models. Some works expect to yield small and accurate deep models by more delicate model design or direct training of promising network architectures rather than iterative execution of training and pruning.

For instance, the authors of [65] propose a small CNN architecture called SqueezeNe and put forward a network architecture similar to inception and call it a "Fire Module", which consists of a squeeze convolutional layer with 1×1 filters and an expanded convolutional layer with both 1×1 and 3×3 filters. It reduces the computation overhead of convolutional layers by reducing the size of filter kernels and the number of channels.

The authors of [66] learn from factorized convolution and divide convolution into two parts. The first part is depthwise convolution, in which each filter only convolutes certain input channels. The other part is pointwise convolution, using a 1×1 filter to combine the multichannel outputs of depthwise convolution layers.

Lin et al. [35] attempt to overthrow the previous idea of network pruning. Their goal is not to prune after training a large model but to design a small model architecture and train it at the beginning. Through extensive evaluation of state-of-the-art pruning algorithms, they found that it is the network structure that counts, rather than retaining some weights.

2.1.2. Lowering time complexity

Time compression of models means lowering the inference time complexity of deep models with no notable increase in the training time complexity. By reducing the time complexity of the deep model, we may be able to reduce the amount of battery power it requires. Common

methods that reduce the time complexity are as follows:

2.1.2.1. Low-rank tensor decomposition. Low-rank tensor decomposition is often based on the theories and methods of low-rank approximation, which is a very useful tool for speeding up large CNNs. It decomposes the original weighted tensor into two or more tensors, and optimizes the decomposed tensor, and it usually uses standard SVD to unfold three-dimensional tensors into two-dimensional tensors, or uses the outer product sum approximation of multiple unidimensional tensors to decrease the time complexity. For instance, Cheng et al. compute the low-rank tensor decomposition to eliminate redundancy in the convolution kernels. Their method obtains a global optimizer of the decomposition. On this basis, they use a new way to train low-rank constrained CNNs from scratch [67].

2.1.2.2. Computation acceleration. Computation acceleration usually speeds forward propagation by improving matrix multiplication, abating numeral resolution, etc. Park's work proposes a tensor-based multiplication to achieve efficient computation between a dense matrix and a sparse matrix. It pre-locates the values that will be zero to avoid calculating them when multiplying matrixes [68].

2.1.2.3. Knowledge distillation. The concept of knowledge distillation was first proposed by Bulica in 2006 [69], and Hinton summarized and developed it in 2014 [70]. The main idea of knowledge distillation is to train a small network model to imitate the knowledge of a large network trained in advance, similar to the relationship of the teacher and the student. The large network is the "teacher", and the small network is the "student". The general method is to learn softmax classification outputs of teacher models. For example, Hinton et al. reduce the amount of computation in a deep learning network by following a less cumbersome model. To transfer the generalization ability of the cumbersome model to a small model, they use the class probabilities produced by the cumbersome model as "soft targets" for training the small model and add a small term to the objective function to encourage the small model to predict the true targets.

2.1.2.4. Data transformation. In this method, the input data is transformed to speed up the calculation. For example, Chen et al. use the idea of wavelet to decompose the original input image into two low-resolution subimages, one of which carries high-frequency information and another carries low-frequency information. In this way, their work reduces image resolution without losing original information and finally achieve deep computation speed [71].

2.1.2.5. Direct design of small models. To reduce the inference latency of mobile deep models, we can directly design small, fast mobile models and ensure their accuracy at the same time. There are various types of MDLA, and each has different trade-offs between model size and accuracy. Manually balancing these trade-offs and designing deep models for each of them are very difficult because there are so many factors to consider. In the most recent work [72], the authors propose an automated Mobile Neural Architecture Search (MNAS) approach that uses deep reinforcement learning to search for a model for a specific trade-off. They also propose a novel factorized hierarchical search space to obtain a good trade-off between flexibility and search space size. Using deep learning and automatic searching to avoid the complexity of manual design is a promising new direction to build mobile deep models. And for these lightweight convolutional neural networks (such as MobileNets) designed for reasoning on mobile devices, Depthwise Convolution (DWConv) and Pointwise Convolution (PWConv) are their key operations. Taking into account the characteristics of current mobile hardware and software systems, Zhang et al. proposed techniques to re-optimize the implementations of these operations based on the ARM architecture [73].

2.2. Reuse of intermediate results

In addition to modifying the structure of deep learning models, it is applicable to reuse the intermediate computing results of deep models by caching part of the computing results based on the partial similarity of the data. As a result, it can decrease the model's computational resource needs. The intermediate results to be reused can be considered at multiple granularities, including the middle layers' computational results for the same model and a similar input, a shared similarity search for different models and the same input, similar semantic computational results for the same models on different devices and a similar input, etc. The basic idea of these techniques is to reduce the computational resources for running MDLAs.

2.2.1. Data reuse among image frames

Data reuse among image frames is often applied to the same model and similar input, typically in continuous mobile vision. It is a serial mobile video image stream obtained from omnipresent cameras on mobile and wearable devices to support diverse vision MDLAs, including recognition assistance, lifestyle monitors, street navigation, etc. The CNN is a state-of-the-art vision processing method, which can be regarded as a group of stacked layers. Each input frame generates intermediate results called a feature map and outputs reasoning results. Because of these characteristics, we can reuse its layer processing results for similar continuous images.

For example, DeepCache discovers temporal locality from the inner structure of the input video, that is, the similarity between frames with similar time. Referencing the heuristic approach in video compression, DeepCache propagates areas of reusable results in frames using the model's inner structure. These inner elements are not pixels. Instead, they are high-dimensional, inexplicable data [6]. In Ref. [29], the authors design an intelligent caching mechanism, especially for convolutional layers. The key idea is to utilize the similarity between successive frames in the first-person video. It computes the current frame by reusing the middle results of the previous frame through the inner structure of the convolution layer rather than simply reusing the final output.

2.2.2. Data reuse among multiple deep tasks

Data reuse among tasks is often implemented for different deep models and the same input. MDLAs might have several models for different but related tasks with the same input executed during a similar time, and it is a waste of resources to repeat the underlying feature extraction calculation. The popular idea is to share partial computing results of models for multiple tasks. For example, an MDLA may aim to infer the race, age and gender of the user. One choice is to train a DNN for each task, which incurs a cost of four inferences. DNNs can be treated as bottom layers for extracting the feature representation and high layers aiming for different tasks. The abstract features computed at the bottom could be used by multiple high layers. Hence, the main idea of data reuse among multiple deep tasks is that feature representations computed by lower layers could be shared among different high layers to save computational cost.

The authors of MCDNN apply the idea above and present "class" clustering, a DNN classifier specialized for similar tasks. They aim to dominate the context and provide similar classes with specialized, light models. More importantly, the model needs to perform well in recognition when the input does not belong to any of the classes and the classifies it as the "other" class. Concatenating this specialized model with a generic model, and only if the specialized model reports the input as the "other" class does the general model perform further classification [36].

2.2.3. Data reuse across devices

Computational data reuse across devices is applied for the same models on different devices and similar inputs. There are many scenarios of running the same MDLA on adjacent multiple devices, and these application cases often process similar contextual data mappings for the

same results. The authors of foggy reuse approximate computation reused inputs across devices, which minimizes redundant computation by harnessing the "equivalence" of similar input values and reusing previous computation outputs with high confidence. They designed Adaptive Locality-Sensitive Hashing (A-LSH) and Homogenized k Nearest Neighbors (H-kNN). The former achieved extensible and constant lookup, while the latter provided high-quality reuse and a tunable accuracy guarantee [37].

2.3. Resource dispatch among multiple tasks

There is usually more than one MDLA running on a mobile system. The joint resource dispatch optimization among all these deep models instead of independent optimization for a single deep task can maximize the sum of the performance of all MDLAs. For instance, the architecture of DeepEye eliminates a crucial limitation of executing multiple deep learning models on resource-limited mobile devices by presenting a novel inference software pipeline. Its goal is to combine the execution of heavy computational convolution layers with high-memory-cost fully connected layers, which realizes partial execution of multiple models simultaneously, especially for CNNs [38].

NestDNN considers that resources available in mobile devices are dynamic due to events such as starting new applications, closing current applications or modifying the application priority. Based on this consideration, the approach presents a multitask resource-aware deep learning framework for mobile vision systems. It selects the optimal resource balance and accuracy for each deep learning model dynamically so that the models'resource needs are compatible with the available resources in the runtime system [5].

Geng et al. solve an energy-saving local core-offloading problem for multiple deep learning tasks running on multicore mobile devices. They formalize the problem as a mixed-integer nonlinear programming problem and then propose a heuristic algorithm to jointly solve the offloading decision and task scheduling problems. This strategy prioritizes tasks of various applications to satisfy both application time constraints and task-dependency requirements. To reduce the search cost, they recursively inspect tasks and move them to the right CPU cores to minimize the energy cost [39].

2.4. Deep learning frameworks on mobile devices

There have been many well-established deep learning frameworks on resource-sufficient computation platforms, such as Caffe, TensorFlow, and Torch. They display good performance in various deep learning tasks and have been widely used. However, for resource-restricted mobile platforms, these heavy frameworks are not appropriate. Firstly, although these frameworks are complete, most of them have third-party dependencies, which makes the framework cumbersome and unsuitable for deployment on resource-constrained mobile devices. Secondly, the mobile deep learning framework should be more focused on the inference phase, which is inconsistent with the traditional framework design. Moreover, there is no model compression or mobile hardware-oriented calculation optimization in a non-mobile deep learning framework, which is exactly the focus of mobile deep computing optimization. Although some frameworks have released corresponding versions for mobile platforms, they are primarily optimized for the powerful GPU chips that facilitate the training process while bringing little profit in the inference phase. Recently, some work has focused on developing professional software libraries to train and deploy deep models on resource-limited mobile devices. The general idea is to combine traditional frameworks and accelerate the inference process of trained networks to significantly reduce the resource of running MDLAs.

For example, DeepLearningKit supports the use of CNNs on mobile devices with GPUs under the IOS system. It can speed up the inference phases of deep models trained by Caffe [74]. DeepX, as a software accelerator, also optimizes the computation, memory, and energy cost of the inference phases of deep networks trained previously. Its method is to divide the network's computation into simpler pieces that can be arranged more efficiently. Each piece can be run on different processors (e.g., GPU, CPU) to achieve sufficient utilization of the computation ability of mobile devices [75].

In Table 1, we list some current state-of-the-art deep learning frameworks on mobile devices.

2.5. Summary of this chapter

This section illustrates the local deployment of MDLAs and presents four ideas: reducing deep model complexity, reusing intermediate computing results, performing resource dispatch among multiple deep tasks and developing deep learning frameworks for mobile systems. In Table 2, we give a summary of this section.

3. Distributed deployment

In addition to local deployment, MDLAs can be executed in a distributed way, especially for computationally intensive applications. The development of communication techniques makes this possible. Offloading is a solution to enhance the capabilities of mobile systems by migrating computing to computable nodes with more resources. With good network connections, we can offload tasks from various MDLAs to resource-sufficient backgrounds and use abundant storage resources in different locations of the network to support its operation.

3.1. Position

3.1.1. Remote cloud

Before mobile-edge computing and D2D communication techniques, people usually offloaded computation to traditional cloud service platforms and introduced Mobile Cloud Computation (MCC). Cloud services deployed on core networks have rich computation and storage resources, enabling the running of more complicated applications on mobile devices and saving devices' resources. In most cases, this approach can be a good choice for MDLAs with small data transference needs, large computation needs and low interaction needs.

For example, Shea et al. provide a platform to offload AR content that requires real-time image analysis and a great deal of image rendering to powerful cloud servers [50]. This kind of AR MDLA is computationally intensive, has a high power cost and is a typical application type to be offloaded to clouds. The authors use Pokemon Go, a popular AR game, to test their platform.

However, the mobile cloud still faces multiple challenges. Obviously, because of its relatively long distance from users, mobile cloud computation is not suitable for all offloading cases, especially for interaction-intensive applications. MCC also places massive additional loads on the radio and backhaul of mobile networks. Dinh et al. list the technical challenges of MCC in detail [81]. In mobile communication, because of the characteristics of wireless networks, e.g., lack of wireless resources, traffic congestion and multi-rates, MCC faces challenges, including low bandwidth, service availability, heterogeneity, etc. Regarding the computational aspect, there are difficulties such as efficient and dynamic computation offloads under variable conditions, user and data security problems, the productivity of data access, and contextual awareness. These are not the focus of this article.

3.1.2. Edge network

Due to the long distance from the user and the limited backhaul bandwidth, it is difficult for MCC to cope well with all offloading scenarios, especially for latency-sensitive and interaction-intensive MDLAs in recent years. Over time, people have continued to explore how to promote resources and services closer to users to reduce access delays and energy consumption. Then, mobile edge computing, ad hoc clouds, and other novel computational architectures emerged. Mobile Edge

Table 1
An overview of the popular deep learning frameworks for mobile terminals.

Name	Company	Android	IOS	CPU	GPU	DSP	Time	Open source	Characteristic	Training
TensorFlow lite [40]	Google	✓	✓	✓	✓		2017	✓	Lightweight, cross-platform, fast	×
Caffe2 [41]	Facebook	✓	✓	✓	✓		2017	✓	Lightweight, modular, scalable	✓
core ML2 [42]	Apple	×	✓	✓	✓		2018	×	Weight quantification, batch forecasting	×
NCNN [43]	Tencent	✓	✓	✓	×		2017	✓	Cross-platform, no third-party dependence, compiler-level optimization is extremely fast and scalable	×
Feather cnn [44]	Tencent	✓	✓	✓	×			✓	Lightweight (hundreds of KB), no third-party dependency, scalable	✓
SNPE [45]	Qualcomm	✓	×	✓	✓	✓		✓	Executes any depth model, greatly limited by hardware	×
MACE [46]	Xiaomi	✓	✓	✓	✓	✓	2018	✓	Heterogeneous acceleration, compiler-level optimization, supports various framework model transformations	×
Paddle model [76]	Baidu	✓	✓	✓	✓	✓	2017	✓	Multi-hardware platform, deep model quantization compression	×

Computing (MEC) is a promising solution to compensate for the MCC problem. It is closer to mobile users by providing IT and cloud computing in wireless access networks [51]. As one of the key techniques in the 5G era, MEC has some advantages, including low latency, high bandwidth, real-time wireless network information, and location awareness. In recent years, there have been many works studying offloading mobile computation to MEC servers.

For instance, Liu et al. study the offloading of computation-intensive Mobile AR (MAR) tasks to servers in edge networks. They design an edge network orchestrator and build a measurement-based analytical model to express the trade-off between latency and accuracy. They also propose a corresponding algorithm as the core component of the orchestrator to perform server assignment and frame resolution. In their edge-based MAR system, mobile tasks are first offloaded to the orchestrator and then sent to the edge server [56].

In recent years, ad hoc mobile cloudlets [57] have emerged as a closer offloading point for mobile users. They can offload their computation to peer devices ad hoc to save local energy and resources. When we use cloudlets, there are more communication methods than MEC and MCC, such as Bluetooth, Wi-Fi Direct, and other direct communication techniques, which may lead to lower latency. Van et al. discuss the optimal offloading decision for mobile users in ad hoc mobile clouds. Mobile users can offload computation to nearby cloudlet devices through Device-to-Device (D2D) communication-powered cellular mobile networks. The authors developed an offloading scheme based on Deep Reinforcement Learning (DRL), aiming to make an optimal offloading policy by considering the uncertainty of users and cloudlets as well as the resource availability of cloudlets [82].

3.1.3. Collaboration of remote cloud and edge networks

MCC offers high-capacity service and rich computing resources, but it is too far from mobile terminals and faces high latency. MEC is closer to mobile users and offers low latency, but it has limited computing resources and low-capacity queries. Ad hoc mobile cloudlets are the closest to mobile users and offer more communication methods and low latency. Users can access cloudlets through one hop in the wireless network, but resources on cloudlets are extremely limited and offer few services. Therefore, when multi-mobile users offload their tasks to cloudlets, resources on cloudlets are likely to be depleted, and the rejection rate of new requests is high. Each type of MDLA has different characteristics, such as latency sensitivity and computation intensity. Therefore, combining MCC, MEC and ad hoc mobile cloudlets, learning from others' strong points and closing the gap would be a promising approach for MDLAs.

For example, Teerapittayanon proposes a Distributed Deep Neural Network (DDNN) based on distributed computing hierarchies, including clouds, edge networks and terminal devices. The DDNN can accommodate DNN inference in the cloud, as well as rapid and local inference at edge servers and terminal devices using shallow parts of the DNN. Under the support of extensively distributed computing hierarchies, a DDNN can extend neural networks and geographic scales [83]. Its distributed method also improves the sensor fusion ability, fault tolerance and data confidentiality of DNNs and can be applied to large MDLAs because of its more robust and safer operation.

In [84], Vitor et al. provide a new strategy to simplify the combination of cloud and fog facilities in IoT scenarios, called the Combined Fog-Cloud (CFC). It introduces the QoS-aware service allocation problem and expresses it as an integer optimization problem to meet capacity requirements and minimize latency.

Lin et al. coordinate the computational resources of cloudlets and remote clouds to fully utilize these two systems. Additionally, they develop a system reward model for wireless bandwidth and computational resource allocation. They formulate the problem as a semi-Markov decision process and use the LP solver tool to solve it as a linear programming problem [53].

3.2. Computation mode

3.2.1. Single-server single-user

Under the condition of the single-server single-user, we mainly focus on making the offloading strategy, that is, running MDLAs locally, performing complete offloading or dividing MDLAs into several independent subtasks to partially offload. Our goal is to minimize energy cost or tasks' total latency or make a trade-off between the two. We discuss the offloading strategy in detail in subsection B.

3.2.2. Single-server multi-users

The single-server multi-user mode is more complex than single-user situations since we must consider issues of computational resource allocation and wireless channel competition. When multiple users offload their computation to one server to reduce the processing time of their MDLAs simultaneously, their response time may not be as perfect as expected. On the one hand, this situation may cause resource contention over limited servers. On the other hand, there will be competition for various resources, including wireless connections. Under such circumstances, to achieve better system performance, joint optimization of offloading strategies of individual users under multiple constraints on their time and energy should be taken into consideration.

Josilo et al. study the resource allocation problem of multiple self-benefiting mobile users offloading to mobile clouds [52]. They define this as a non-cooperative game problem and present an efficient decentralized algorithm to jointly optimize their offloading strategies. Their algorithm converges to a pure-strategy Nash equilibrium. Finally, an upper bound for the price of anarchy in the game is provided for the two cloud resource models they propose.

Liu et al. introduce a joint multi-resource allocation framework located in a cloud computing system based on the Semi-Markov Decision Process (SMDP). The goal of the framework is to maximize the overall

Table 2
Overview of work for local deployment of MDLAs.

Key idea	Bibliography list	Work content
Decrease the complexity of the deep learning model and make it suitable for mobile execution	[30]	The unimportant connections in the network are pruned after training to reduce the parameters needed by the network and the consumption of memory and CPU
	[31]	Using a particle filter, the importance weight of each particle is allocated by calculating the error classification rate with corresponding connected modes
	[77]	The most advanced Triple-Response Residual (TRR) method is used to sort the filters in a given deep learning model according to the importance of the filters as well as to prune the filters iteratively
	[32]	Output units are maximized and multiple neurons are merged into more complex convex function representations
	[33]	The parameters are mapped to the corresponding Hasi bucket, where parameters in the same bucket have the same value
	[34]	The weight and input of CNN network layers are binarized so that the computing speed is faster and the memory consumption is smaller
	[35]	The original pruning point of view is overturned and the acquired network architecture rather than the retained weight is treated as more important
	[68]	A multiplication method based on vector form is used, and the values that will be 0 are locked in advance;
Reuse intermediate results	[78]	The internal structure of the model is used to propagate areas with reusable results in the matrix
	[50]	An intelligent caching mechanism is designed for the convolution layers, which makes use of the similarity between consecutive frames in the first-person video
	[37]	Computing results are reused across devices between similar applications
	[51]	The idea of "class" clustering is used to specialize a DNN classifier for similar classes. The specialized model is connected with the original "generalized model" variant of the model
Resource call between multiple deep learning tasks on a single mobile device	[79]	A new reasoning software pipeline is proposed, which aims to interweave the execution of the convolution layer with a large amount of computation and the loading of the full connection layer with a large amount of memory in order to realize the local execution of multiple depth vision models (especially CNNs)
	[77]	A framework is proposed that considers the dynamic changes of runtime resources to realize deep learning for resource-aware multi-tenant devices in a mobile vision system
	[80]	The aim is to solve the problem of energy-saving computational offloading of multiple deep learning tasks running on multicore mobile devices
Mobile deep learning framework	[40–46,76]	
Automatic design of small models	[72]	Deep reinforcement learning is used to search for mobile models, which has a good trade-off between accuracy and latency to reduce inference latency

advantages of the entire system by constructing an optimal strategy of wireless bandwidth and computing resource allocation for multiple mobile users in MCC that has a low service-denial rate and processing latency [53].

Zheng et al. adopt a multi-user stochastic game-theoretic approach in an MCC dynamic offloading environment. Mobile users are in a dynamic state, active or inactive, and radio channels vary stochastically. The authors formulate an offloading strategy for multiple users in a dynamic environment as a stochastic game due to its selfish character and prove that it is equivalent to a weighted potential strategy with at least one Nash Equilibrium (NE). They propose a multiagent stochastic learning algorithm with convergent speed to solve the problem [54].

3.2.3. Multi-servers single-users

In this case, we first set the premise that the server is not a remote cloud; it is a set of edge servers or computable devices not far from mobile devices running MDLAs. When considering different offloading modes of MDLAs, the multi-server single-user mode can be divided into two situations: (i) the MDLAs are offloaded completely, which means that the mobile device needs to choose one point among multiple computable locations according to the current wireless channel state, task queue and available resources on edge servers and nearby computable devices. It is defined as a request routing problem, and we discuss it in paragraph D; (ii) MDLAs can also be divided into several subtasks and offloaded to multiple edge servers and computable devices. In this case, the heterogeneity and time-varying nature of edge devices pose difficult challenges for the division and allocation of tasks and the collection and consolidation of calculation results. Keshtkarjahromi et al. propose a Coding Cooperative Computing Protocol (C3P). It dynamically offloads encoded subtasks of MDLAs to multiple computable locations in the edge network and can adapt to time-varying edge resources [85].

3.2.4. Multi-servers multi-users

Under multi-server multi-user conditions, there are more problems to be noted. We define them as follows:

Resource allocation on a single server: Resources on edge servers are limited and may fail to satisfy all requests from the covered area. We define the allocation of limited resources on servers among multiple users as a resource allocation problem.

Users' routing requests on multiple servers: The density of base stations in the 5G era will reach 50 BSs/km^2 [86], which will lead to users being located in a complex multi-base station environment with overlapping areas. Such complicated, multi-unit situations make it difficult for users to decide where to offload their MDLA tasks to achieve the optimal performance of MDLAs, and we define this as a request routing problem.

Placement of service: Services can be cached on edge servers not far from users to provide lower service access latency. There are three prerequisites: (i) edge servers can only cache a limited number of services; (ii) users' requests can only be routed to servers with the service they request; and (iii) users are in a complex multi-unit environment. We must decide how to allocate various services among multiple edge servers to respond to more requests and maximize overall performance. We define this as a service placement problem and discuss its solution in subsection C.

Balancing offloading among edge servers: The distribution of mobile users presents great spatial variety and mobility. These characteristics cause an imbalance in the workload among edge servers and influence the request-response time. We can define two problems from this: a load-balancing problem and a resource-sharing problem.

With the definitions of these problems, we can now provide some examples of offloading under multi-server multi-user modes. Each example faces one or more of the problems above.

Poularakis et al. formulate the Joint Service Placement and Request

Routing problem (JSPRR) in a multi-unit MEC network. Its optimization target is to minimize the amount of centralized offloading to the cloud caused by a lack of service caching or insufficient resources at the edge. Then, they propose a bicriteria algorithm that provably achieves approximation guarantees while violating resource constraints in a bounded way [87].

Liu et al. primarily focus on users' request-routing problems. They study the dynamic allocation of users' offloading requests under multiple edge servers in a MAR system [56].

Chen's work focuses on resource allocation and request-routing problems. It studies offloading in a superdense computing network based on the idea that the software defines the network. The authors transfer the offloading strategy as an NP-hard mixed integer nonlinear programming problem and further divide it into two subproblems: the convex subproblem of resource allocation and the 0–1 program subproblem of request routing. They use alternative optimization techniques to find a solution [88].

The work of [89] mainly focuses on the resource-allocation problem and resource sharing among multi-edge computing servers. It models these problems as a multi-objective optimization problem and constructs a framework based on Cooperative Game Theory (CGT) in which every edge server first satisfies its own offloading request and then shares the remaining resources with other servers. The authors present an O(N) algorithm and achieve Pareto optimal allocation.

3.3. Offloading decisions

Offloading the tasks of various MDLAs to backgrounds with sufficient resources is the basic idea of the distributed deployment of MDLAs. When we make an offloading decision, a key challenge is to decide when and how to offload, because offloading is not always beneficial; unstable network conditions, frequent interactions or large amounts of input data may lead to large transmission latency and high energy consumption. However, making the best offloading decision is not an easy and straightforward task. For different MDLA types, there are different factors to be considered and different weights of demands, including accuracy, latency, energy consumption, etc. We also need to consider the state of the whole system, including the device temperature, current task number, network type, state of the background server, etc. The inherent complexity and diversity of these factors have led to a variety of studies on computing offloading decision-making. For the offloading mode, computing offloading can be divided into two strategies: complete offloading and partial offloading. Decision-making methods can be divided into rule-based and learning-based methods.

3.3.1. Offloading mode

A typical MDLA can be simply divided into three parts: data acquisition, data preprocessing and data analysis. Data acquisition often requires hardware to be integrated into mobile devices. Therefore, due to the limitation of hardware settings, it must stay on mobile devices and cannot be offloaded. For the other two subtasks, the optimal offloading decision should be made under the premise of comprehensively considering the resources needed, the amount of communication data between subtasks, battery power and the current network bandwidth. In addition, it should be noted that during partial offloading, the order of offloading among subtasks is reversed to the running order.

3.3.1.1. Complete offloading. Highly integrated or relatively simple tasks cannot be partitioned and must be executed locally or offloaded as a whole to a server or peer-to-peer device. Alternatively, MDLAs can be partitioned, but after making offloading decisions, complete offloading is considered to be the current optimal solution. In Ref. [58], the authors define such a computing task model as a binary offloading task model and use three field symbols to represent its properties: the task input data size, time limitation and calculation workload. These three features are the basic attributes of an MDLA, and we can essentially use them and the current network bandwidth dynamics to make offloading decisions.

In a study [59], Pavel Mach and Zdenek Becvar investigate complete offloading from three perspectives: minimizing latency, minimizing energy consumption under delay constraints, and trading off delay and energy consumption. In another project [50], the authors present a platform for offloading MAR tasks to powerful cloud servers completely. They implement this system using a thin-client design.

3.3.1.2. Partial offloading. An MDLA consists of many components and can be divided into multiple partitions to achieve fine-grained (partial) computing offload.

In partial offloading, we must note the following three points: (i) The dependence of partitions and components influences the executive order of components and the offloading sequence. The components higher in the executive order have a lower offloading priority. (ii) Hardware-constrained components must be executed locally on mobile devices; for example, in a mobile video analysis application, we obtain an image or video stream through a camera on the mobile device that cannot be offloaded. (iii) The size of data exchanged between components and the amount of computation of each component should also be considered. The tendency is usually to offload components with a large amount of computation or less data traffic with other components or offload in the reverse order of execution.

We may consider three aspects of potential strategies: (i) offloading some subtasks of the MDLA to the background to reduce the calculation delay and energy consumption of mobile devices; (ii) offloading part of the processed data instead of all initial data to reduce the transmission delay and energy consumption; and (iii) protecting the security and privacy of user mobile data. We give some examples of previous work to illustrate each aspect.

For point a, the work in Ref. [18] aims to detect a specific object in videos on mobile devices. It can be divided into three subtasks: video capture, video frame extraction and frame detection. Video capture can only be done by cameras on mobile devices and must be performed on mobile devices locally because of hardware limitations. Video frame extraction and frame detection can be offloaded sequentially according to network connection conditions and battery power. Notably, the order of offloading of these two subtasks is limited. The first subtask to be executed is the last subtask to be offloaded.

For point b, Jain et al. aim to use environmental fingerprinting to achieve immersive, highly contextualized MDLAs, especially MAR. This visual diversity requires matching match a unique visual signature with millions of databases. Its computation is heavy, and it needs to offload considerable visual data to the cloud. The authors identify the low-entropy characteristics of visual "features" and design a system named VisualPrint to offload only the most distinctive visual data, reducing the time of network transmission [49].

For point c, when offloading the data of an MDLA to the background for better and faster execution, mobile users face the risk of data privacy exposure. Partial offloading is beneficial to privacy protection in data exchange [20,22,23]. Intermediate data in deep learning models usually have semantics different from those of row data. For example, it is difficult to understand original information by only observing the features extracted from the original data by CNN filters. Therefore, we can offload high layers of the deep learning model, and then offload abstract data processed at the bottom layer of the mobile side to the background. In many works of distributed deep learning, the model on mobile devices is regarded as a worker and is combined with a central server to train the whole deep model. Partial local computing abstracts the user's private data to a certain extent and protects data privacy and safety when it is offloaded to the central server. This can also be applied in a similar way to most mobile crowdsourcing scenarios.

3.3.2. Decision method

3.3.2.1. Rule-based offloading decisions. A rule-based offloading decision usually considers whether to offload as the output result of a combinatorial optimization problem under a set of constraints. This method formulates the problem by measuring the context of current task execution under specific constraints and optimizing objectives, and uses certain mathematical knowledge to solve the problem and output the decision scheme.

In the field of vision, we list some works of rule-based offloading decisions. In Refs. [90,91], Ran et al. make extensive measurements to understand the trade-offs between video quality, network conditions, battery consumption, processing delay, and model accuracy, and formulate them as an optimization problem; then, they use a measurement-driven mathematical framework to efficiently solve this combinatorial optimization problem.

In [18], Lu focuses on mobile video analysis; the task publisher needs mobile crowdsourcing videos to identify specific objects. A video crowd processing platform is designed and offloading decisions are made in both Wi-Fi and mobile cellular network connections. Under Wi-Fi connection, the optimal goal is minimizing completion time, and an algorithm named split-shift is proposed. Under cellular connections subjected to data usage constraints, the optimization goal is the trade-off between processing time and energy consumption.

The authors of [56] consider dynamically assigning the workload of the mobile AR system to multiple mobile edge servers to maximize the performance of the MAR system. Liu et al. formulate the trade-off of network latency, computational latency, and analytic accuracy in MAR systems and develop a multi-objective optimization problem to select the optimal edge server and video frame resolution for MAR users. They design a fast and accurate (FACT) algorithm to solve this multi-objective optimization problem based on convex optimization theory.

In other fields, for offloading of one MDLA, the following works make rule-based offloading decisions. Sundar et al. study offloading decisions of MDLA in terms of a set of dependent tasks in a general cloud computing system consisting of a heterogeneous local processor network and a remote cloud server. Their optimization target is to minimize the execution cost of the entire application under each subtask completion time constraint. They propose a heuristic algorithm named ITAGs to solve this NP-hard problem [80].

The work of [92] aims at minimizing task delay. This optimization problem takes the queue state of the task buffer, the execution state of local processing units and the state of transmission units as inputs to determine whether to offload completely.

Xu, Chen and Zhou regard the minimization of the computational delay and device energy consumption on the server as the optimal target, and their constraint condition is the cache capacity of the edge server, the maximum delay limitation of tasks, and the battery power of the device. Their system outputs a service placement layout on servers and an offloading decision for devices.

Third, for offloading multiple MDLAs on multiple mobile devices, we list the following works, most of which jointly optimize the offloading decision of these tasks. The authors of [93–95] study the joint optimization problem of multi-task offloading under multiple mobile devices. They measure the arrival rate of the data packet of each time slot and the current network channel conditions as input, use the complete time limitation of each task as a set of constraints, and aim to minimize the energy consumption of these mobile devices. They finally output the offloading decision for each mobile device and the allocation of wireless resources and computing resources on the server among multiple tasks.

The work of [96] also considers the joint optimization of multitasks on multi-mobile devices. It minimizes the trade-off between the energy consumption of mobile devices and task execution latency and the output offloading decision for each task and its optimal wireless channel selection.

3.3.2.2. Learning-based offloading decisions. Making offloading decisions based on learning begins by collecting, quantifying and characterizing the current running context of the program, including battery power, application properties, user mobility, network status, etc., and then uses them as input to the deep learning model to output whether and how to offload at the current moment. At present, learning-based methods mainly use two basic strategies: DNNs, which are usually used to construct classifiers, and DRL, which has an excellent performance in decision-making.

In [97], the authors propose a novel mechanism for optimizing offloading performance by using crowd-sensed evidence traces and constructing a DNN offloading-decision classifier. They believe that for MDLAs, data from one device is obviously not enough to quantify individual factors affecting offloading due to their inherent complexity and diversity. Huber Flores et al. aggregate samples from a larger community of devices and design an evidence analyzer using a DNN to identify times beneficial for offloading by analyzing evidence traces collected through crowdsensing.

Duc Van Le et al. propose a Deep Reinforcement Learning (DRL)-based offloading scheme that enables users to make near-optimal offloading decisions under consideration of uncertainties of user and cloudlet movements and cloudlet resource availabilities. They propose a Markov Decision Process (MDP)-based offloading problem formulation and then use a deep reinforcement learning scheme called a Deep Q-Network (DQN) to learn an efficient solution for the proposed MDP-based offloading problem [82].

In addition to making an appropriate offloading decision, we can improve the effectiveness of offloading by special offloading means. For example, Wasiur et al. use a queuing theoretic description of a collaborative uploading scenario, split data into chunks and offload them over multiple paths; finally, these chunks are merged at the destination [98]. This method can reduce the network transmission delay significantly and can be generally applied to other offloading work. This approach is a special offloading technique rather than a black box that generally uses the network state as input and outputs an offloading decision. We can also consider other special offloading methods for each field, although this may be unusual and not suitable for general work; however, for offloading tasks in certain fields, compared with the conventional method, it is a good way to further improve the performance of offloading.

3.4. Distributed cache

In the broad use of MDLAs, we can observe two points. First, a user-requested service has a high degree of repeatability; that is, the same application is downloaded and run on thousands of different mobile devices by thousands of users in the application store. By deploying these services in a mobile edge network, mobile users can easily offload their MDLA data to edge servers under good network conditions, which can greatly reduce the MDLA's execution latency. Whether the service can be cached in a certain edge server directly determines whether the user in its coverage region can offload their computing to the edge. Second, in MAR applications and many other video image MDLAs, similar video content may be repeatedly requested by many users. Because the video transmission takes up a large bandwidth, if we take the video from the CDN for each request, the repeated content transmission will cause great bandwidth waste. Therefore, we should adopt an intelligent cache strategy in a mobile network to enable mobile users to obtain the content from a nearby cache, which could significantly reduce the data accessing time of the MDLA and greatly eliminate the influence of the network connection dynamics. It has been proved that caches in 3G mobile networks and 4G LTE networks can reduce mobile traffic by 1/3 to 2/3 [99]. In addition, the energy efficiency of the 4G network can be improved. The evolution of the green 5G network can be effectively promoted by the intelligent caching of popular content to reduce traffic load.

3.4.1. Cache content

For MDLAs, there are two kinds of content to be cached at the edge network: deep models of MDLAs, which we call services, and the MDLA input data. Caching services requested by a large number of users on the edge network enable mobile users to offload their corresponding computation, and the benefit depends on the popularity of the cached services. MDLA input data is generally data types with large transmission bandwidths, such as videos, images and common data.

3.4.1.1. Service. A service cache is a deep model of the cache in an edge server or nearby computable device and its associated databases, which allows users to offload the corresponding computing tasks on the edge. Since we can only cache a limited number of MDLA services in resource-constrained edge servers at the same time, we must carefully decide which services to cache to maximize the profit of offloading for the overall system. The services that we cache on the edge server decide which tasks the user can offload. If we cache the most popular service, the system may obtain the maximum performance benefit.

Xu et al. formalize the joint service caching and task-offloading problem in MEC-enabled dense cellular networks to minimize computational latency under a long-term energy consumption constraint. They develop a novel online algorithm named OREO to perform stochastic service caching online without the need for future information [55].

Wang et al. focus on mobile VR applications with the support of online social networks. They divide VR applications into two parts: service entities on servers and client entities on mobile devices. They define the Edge Service Entity Placement problem (ESEP) as the problem of deciding where to place the SE of each user among the edge servers to maximize the economic profits of the edge servers as well as achieve the desired level of QoS for users, and they propose an iterative algorithm named ITEM to solve this problem [100].

3.4.1.2. Data. The input data of MDLAs can be divided into two types: (i) real-time data acquired by hardware on mobile devices such as images for object classification; and (ii) offline data acquired from the content provider in the central storage center, for example, multimedia data to support VR/AR and panoramic views. Distributed data caching works only for offline data, not for real-time data. Therefore, for real-time data, we discuss data compression and transmission; for offline data, we discuss distributed caching.

First, for real-time data, the operation is offloading, and the main problem is limited dynamic wireless bandwidth. In addition, we have to face the reality that most depth models are very sensitive to data noise [101], so we need to offload high-quality data. Xie and Kim developed a DNN-aware basic data compression framework named "GRACE" to compress the real-time image and video data acquired on IoT devices, which reduces the network bandwidth consumption for data transmission without affecting the performance of DNN interference on edge servers [102].

Second, for non-real-time data, in many distributed deployment MDLAs, users need to exchange data frequently with the server, or many users may request the same multimedia content from the CDN repeatedly. In traditional cloud-based architecture, content is usually obtained from the central data storage center, far away from users, which is not suitable for the characteristics of frequent data exchange and a large number of access requests. It produces considerable delay and influences users' QoS. Therefore, in recent years, it has become increasingly popular to cache data at places close to users, such as edge servers or ad hoc devices that are nearby.

Zhang et al. propose that more bandwidth is required for VR video applications to achieve high temporal and spatial fidelity content. They design a VR video delivery system based on Named Data Networking (NDN) and propose an integrating hotspot-based and popularity-based caching policy to cache the content that is most likely to be requested to reduce the transmission delay of VR videos and enhance user experience [103].

Hao et al. study knowledge-centric proactive edge caching in mobile content distribution networks. The high dynamics of mobile video streams and complex user playback behaviors make it difficult to decide which content should be cached through popularity-based investigations or probability-based predictions. This work optimizes the caching configuration based on semantic information of the online playback behavior of 5G multimedia service users. They mathematically formulate this NP-complete caching optimization problem and propose a greedy-based online caching configuration algorithm to minimize the overall delivery cost of video streaming and the maximum edge caching utilization ratio [104].

Mohan et al. propose an efficient edge caching mechanism leveraging edge resources to predict and store data required for upcoming computations. Their solution is to group caches according to the workloads of different services. They further develop methods for populating caches and ensuring the coherence of cached data [105].

3.4.2. Cache location

From ad hoc devices to remote clouds, there are many cache locations. Considering the characteristics of the all-IP-based cellular network, we can divide them into three main storage locations: EPC core networks, Radio Access Networks (RANs) and ad hoc devices. However, when deciding the caching location, we need to consider a basic problem: although a closer cache reduces the redundant transmission of identical content in the rest of the network and releases the core, most networks are organized according to a tree distribution hierarchy. And the closer the caching location is to the user, the fewer users it covers. The number of users of the edge server service is less than that of the central cloud storage service. If no user requests data at a certain cache point, it may not be necessary to actively bring replicated content to the edge point. Therefore, intelligent selection of cache locations and optimization of content placement are required.

In Fig. 2, we show the transmission of content in four cases: no-cache, using a core network cache, using the wireless access network cache and using an ad hoc device cache with D2D communication.

In the case of "no-cache", every content request needs to be transmitted through a complex network to retrieve content from a remote ISP, resulting in great storage redundancy and transmission delays. After adding the core network cache, the communication between the ISP and the core network device can be reduced somewhat; after adding the RAN cache, the communication traffic between the access network and the core network can be significantly reduced; if the device cache and D2D communication are further added, the transmission delay can be further reduced.

3.4.2.1. Core network. The centralized deployment of cache servers and PC nodes in EPC greatly simplifies the management of mobile content distribution networks and is easy to expand as content demand diversity increases. The hit rate of the EPC cache can be extremely high because of its massive repository and mass users. Moreover, since the content is encapsulated and transmitted by the GPRS Tunneling Protocol (GTP) downstream of the EPC, it is easier to deploy a content-aware cache on the EPC than in the other locations [99].

However, the core network cache has two main shortcomings: (i) the core network is far from users, which will result in great latency when users access data; and (ii) the core network covers a large range of users, and multiple users request the same content repeatedly, which will waste more network transmission resources.

3.4.2.2. RAN. Although deploying a cache in a core network is easier, recently, many works have preferred pushing content cache to edges closer to users. Especially in the emerging 5G network, Base Stations (BS) are naturally equipped with edge servers (such as Nvidia Jetson TX21) and provide storage capabilities for cache services. By caching

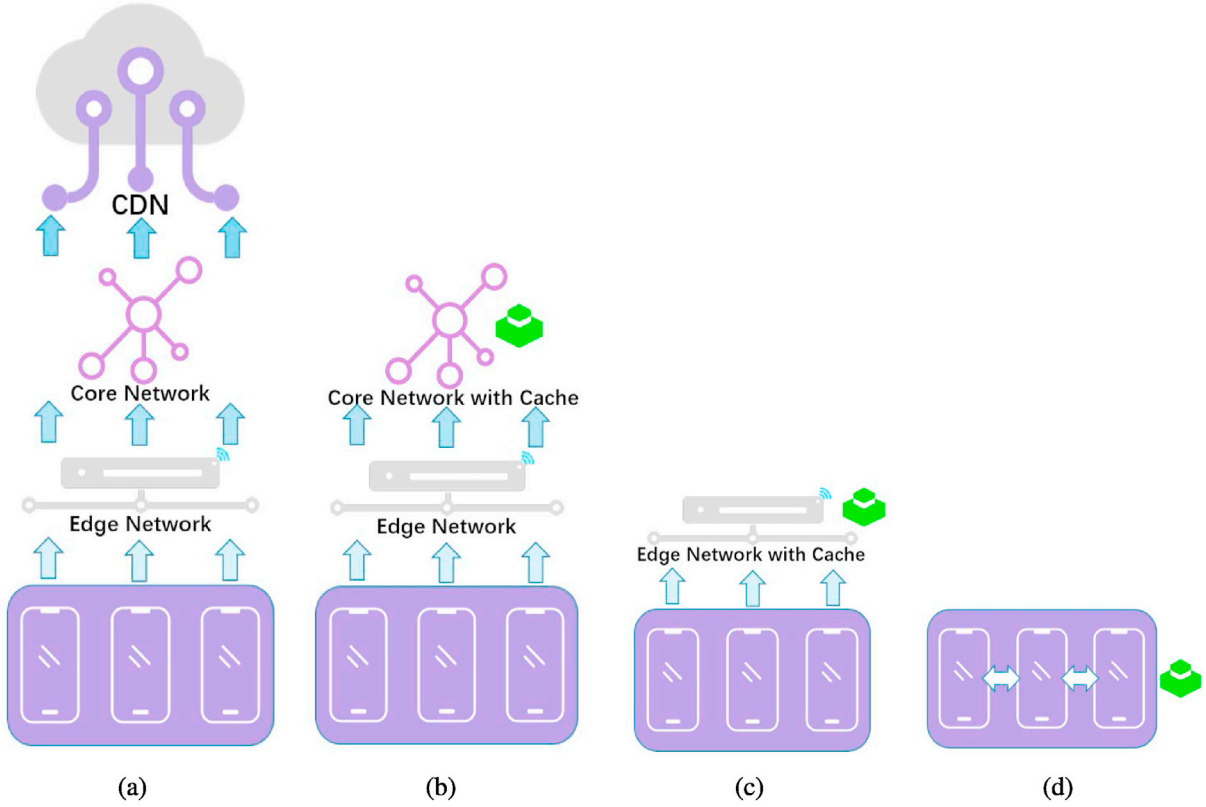

Fig. 2. Different content transfer requests due to different cache locations. From left to right: No-cache, Core network caching, Edge network caching, Ad hoc cloudlet and D2D link.

appropriate content at the nearby edge, viewers can obtain the target content locally instead of from a remote CDN server, which not only provides better QoE with lower latency, but also saves core network traffic costs. RAN cache usually caches content in the eNB, and it is mainly divided into two categories:

Macro base station: a macro base station has a large coverage to serve more users and has rich storage resources for a better cache hit ratio. In the work of Gu [106], the authors analyze the caching distribution problem in a macro base station as an NP-hard problem and propose a heuristic algorithm to solve it.

Micro base station: compared with a macro base station, a micro base station has less storage space and a smaller coverage, so it may have a lower cache hit ratio in terms of the diversity of cache content. However, micro base stations bring greater flexibility, and more importantly, cooperative content sharing between micro base stations can jointly optimize users' requests to improve the cache hit rate. In addition, one of the greatest advantages of micro base stations over macro base stations is that they are closer to the user, so they can bring smaller delays to reduce the impact of unstable network connections.

However, current large-scale content tracking analysis shows that, unlike CDN-based caching, the edge caching environment has a large number of dynamic and diversified request patterns, which is more complicated. Therefore, what content is stored at each base station and how the base stations can coordinate storage to meet the needs of more users is an important issue in this subfield. In the latest work [107], Wang et al. propose MacoCache, which uses Multi-Agents Deep Reinforcement Learning (MADRL), in which each edge can combine with other edges to adaptively learn its own best strategy for intelligent caching.

3.4.2.3. Ad hoc devices. The emergence of D2D communication technology makes it possible to carry out convenient and direct point-to-point communication between terminal devices. It is common to cache content on devices that are geographically close auxiliary nodes or belong to users with similar characteristics and use D2D communication to assist its transmission.

Aravindh Ramane et al. cache data on family auxiliary nodes or mobile devices of social-related users and then connect them together in a distributed way to realize content sharing. They design an edge-caching architecture named "Wi-Stitch", which is an "edge-stitching" distributed content transmission network [108]. In their recent work [109], the Wi-Stitch is extended by solving two main problems: (i) the shared node may not have enough bandwidth to share content associated with the limited Wi-Fi AP; and (ii) Wi-Stitch may produce multiple copies of popular content but insufficient copies of less popular content. The authors formulate these as optimization problems and solve them by strategically placing content for sharing within a geographically localized cell.

Akshay Mete and Sharayu Moharir combine a central server with multiple end-users. They form a content delivery system that supports three content delivery modes: (i) the central server stores the entire content catalog and delivers the requested content to mobile users; (ii) mobile devices have a limited cache and can transmit content to each other through D2D communication; and (iii) the central server transmits the content to mobile users and then transmits it to other users. Their goal is to decide which content to cache on the end devices to minimize service cost [110].

Zhang et al. consider the QoS of a two-hop wireless connection with a delay constraint in a multimedia big data offloading architecture. When two mobile users request the same multimedia data content, one user downloads the data from a BS and uses D2D to forward the data to another. The authors propose three optimal single-hop transmission power allocation schemes to solve the problem of supporting this double-hop wireless link transmission while ensuring the bounded QoS requirements of two single hops [111].

3.4.3. Cache policy

The caching strategy determines the caching content and locations as well as the time when to release its storage. Making a perfect cache scheme is one of the keys to improving the performance of MDLAs. It is important to estimate the benefits of caching certain content by evaluating its current popularity, potential popularity, storage size, and the location of its existing copies in the network topology. It is more challenging than traditional cache strategies, including LRU, LFU, FIFO, etc., and its goal is to improve the cache hit rate and reduce the network transmission bandwidth consumption.

3.4.3.1. Popularity-based. Caching content with higher popularity can maximize the total QoE of all users within a certain region. The popularity of content is defined as the ratio of the number of requests for specific content to the total number of user requests. It is restricted to a certain area within a certain period [112]. However, it is worth noting that the popularity of content is not static; it follows a Zipf distribution, which is a power-law distribution [113]. Therefore, it is necessary to update the popularity of content in a popularity-based cache strategy.

Zhang et al. design a data structure for recording the content popularity of each router. In addition, each router needs to communicate with others to calculate global popularity information [103]. The work of [114] records the global popularity of every video segment at every viewpoint. Large popularity means that this video segment has been requested many times by users, so the cache of the segment is even more meaningful.

The work of [115] analyzes the dynamic adaptability of popularity in the cache algorithm from two aspects: (i) learning the accuracy of fixed popularity distribution; and (ii) learning the changing speed of popularity for certain content. Based on both of these aspects, they propose a novel hybrid algorithm to learn popularity changes faster and better.

Another important fact is that content popularity distribution in a large area is often different from the distribution in a small area. Therefore, the measurement of content popularity faces the difficulty of spatial granularity knowledge, not only because the coverage of various types of edge servers is different but also because the users are in dynamic flow between multiple units. This will also have an impact on the prediction of content popularity, especially for edge servers in SBs with a small coverage.

3.4.3.2. Hot spot-based. In many multimedia MDLAs such as VR/AR, video streaming services, and real-time interactive games, the user usually looks at the most attractive viewpoint. For example, in a football game, the viewpoint of users moves with the movement of the football. In the game, most players are only concerned about their own situation. These areas of interest, or video clips, are defined as hot spots and should be prioritized in caching. Sometimes, there are multiple hot spots in a view. For example, the work of [103] models the attraction of all viewpoints in a VR view to cache the content that is most likely to be requested.

3.4.3.3. User preference-based. The user preference profile includes the probabilities of a specific user requesting each content over a certain period of time, and there are significant differences among different individuals. This is because users usually have a strong preference for specific content categories. Users' preferences can be predicted according to the historical content requirements of users and the similarity between users. This information can be widely used in recommendation systems. Because of the character of preference, user numbers under a certain preference category will not be too large, so it is suitable to be applied to cache servers with small coverage, such as SBs or home cloudlets.

3.4.3.4. Learning-based. First, content popularity is region-specific and not fixed, so it is difficult to capture. Second, in most cases, the content we cache is a video stream that faces highly dynamic and complex user playback behavior. Therefore, a learning-based caching policy using knowledge of content demand history is very promising. For instance, in the work of [78], the authors use multiagent reinforcement learning to design content cache policy in mobile D2D networks without the need to require real-time requirements and popularity. They propose a belief-based Modified Combinatorial Upper Confidence Bound (MCUCB) algorithm to solve the problem of large joint action.

Hao et al. implement a cache policy based on user playback behavior [104]. They use deep belief networks to capture the semantic information of users and infer the video that will be requested in the future based on the user's playback mode. The video is actively cached in the edge network.

3.5. Summary of this chapter

This section investigates the distributed deployment scheme of MDLAs from three perspectives: deployment architecture, offloading decisions and distributed caches. The main work is summarized in Table 3.

4. Typical applications

We provide a brief overview of new MDLAs. First, according to the application scenario or the main beneficiaries, we divide them into user-oriented and third party-oriented MDLAs, including service providers and network operators.

User-oriented MDLAs operate mainly to provide diversified and personalized intelligent services for mobile users, and the user is the main beneficiary. They use lightweight portable data collection devices integrated on mobile devices, including cameras, microphones, sensors, etc., and use these data to compute on nearby mobile devices or edge servers. If the calculation is complex, it can be further offloaded to the remote cloud. This deep learning is lightweight, portable, and close to users. User-oriented MDLAs are user-centric. For example, the real-time video or image acquired by a mobile-side camera provides support for deploying object recognition and tracking technology on the mobile terminal so that the mobile device has vision, which can be used in tasks such as face recognition for user authentication and local video analysis. We can use the camera on a mobile device to collect lip motion video, and the lip-reading information can be used not only for deaf and mute information input but also for lip-reading authentication [3]. Augmented reality and virtual reality technology provide a virtual superhuman vision for us, and it has attracted much interest from academia and industry. It has appeared in many relevant emerging applications, such as remote conferencing, which provides an immersive experience through mobile devices. This technology frees people from a specific space so that they can meet at any time. Various sensors provided on a mobile terminal enable the mobile device to intelligently have a range of awareness similar to humans, such as an acceleration sensor, which gives mobile devices a motion-awareness capability. For example, the data collected can be used to recognize human actions, which can be applied in monitoring human activities [8,118] and in medical scenarios [119]. Another work for human health [12] uses sensors to collect spectral information and design intelligent tableware named Smart-U, which can recognize food composition and analyze dietary information, driving progress in the healthcare system. There are MDLAs that are beneficial for users in many other aspects, such as accelerated browsers on the mobile side [7], mobile malware detection and application recommendation [2], recommendation systems based on historical information and other smart applications [11–14]. MDLAs completely change our lives.

Third party-oriented MDLAs mainly benefit from service providers and network operators. The wide application of MDLAs heralds the generation of massive data. The question of how third parties can effectively use these data to find valuable information and develop better services and new business opportunities is very important. Some typical scenarios are as follows. (i) Mobile crowdsourcing [18–21]: For instance,

Table 3
Overview of distributed deployment of AI applications on mobile devices.

Specific content		Classification	Paper List	Note
Computing	Offloading Policy	Learning-based	[82,97]	Collects running context as model input to auto-output offloading decisions
		Rule-based	[18,56,80,90, 92,94–96]	Takes the minimum time delay and/or the energy consumption of the mobile terminal as the optimization target, under a set of constraints, and formulates and solves an optimization problem
	Offloading content	Reducing transmission	[49,90,98]	By reducing the quantity of data that needs to be transferred
		Protecting data privacy	[20,22,23]	Uploads pre-processed data
	Offloading location	Remote cloud	[50,81]	Remote cloud networks are rich in resources but have large transmission delays
		Edge network	[56,82,116]	Network resources are not as great as those of the cloud but are closer to users
		Collaboration of Cloud and Edge	[83,84]	Combines the edge network with the core network, makes use of the advantages of both, and realizes complementary disadvantages
	Offloading mode	SS-SU	3.2.2	Offloads decision-making on a single device
		SS-MU	[52–54]	Server resource allocation and radio channel contention issues
		MS-SU	[85]	Users have to choose where to offload their tasks among multiple servers
		MS-MU	[56,88,117]	Server resource allocation, wireless channel contention, user request routing, service entity placement, load balancing between servers and resource sharing
Cache	Cache content	Services	[55,100]	Caching app services and related databases in edge servers
		Data	[103,105]	Caching data frequently requested by users, particularly for video
	Cache locations	Core network	[99]	Caching at the cloud with rich memory,but with far distance, and large transmission delay
		RAN	[99,106]	Caching at the eNB or edge servers nearby, leading to faster response speed and lower latency
		Peer devices	[108,110,111]	Caching data closer users,further reducing transmission waiting
	Cache policy	Popularity	[103,112–115]	Uses the probability of certain content being requested in a certain period of time, which is related to the specific region. The cache of content with greater popularity is more important
		Hot spot	[103]	The highest-interest point of users in images or videos; caching high-quality hot-spot content can improve users' experience
		Preferences of users		The user typically has a strong preference for a particular content category; we need to predict and cache the content that individuals prefer
		Learning	[104]	Learning-based caching strategy with content-requesting history knowledge, which has high dynamic adaptability

criminal tracking analysis crowdsources multimedia data to locate criminals; traffic forecasting collects complaints from a large number of mobile users about accidents and congestion on the ground in the early peak period so that service providers can provide more accurate real-time traffic condition reports and obtain better economic benefits. (ii) Distributed deep learning tasks [22–24]: Distributed deep learning on mobile devices to make more efficient and convenient use of the large quantity of data generated by mobile terminals not only solves the problem of large data sets and large models in the traditional centralized training mode but also effectively aims at the problem of data privacy of mobile terminal users. (iii) There are also various internet of things applications [26] and applications that use mobile big data to develop various services [27,28].

5. Future directions

To better promote the development of MDLAs, we summarize some possible development directions based on the following points:

5.1. Mobile devices

As the carrier of MDLAs, mobile devices should be improved in many aspects, such as more comprehensive information acquisition, stronger power of processing units with larger storage, and greater endurance.

5.1.1. Mobile information acquisition device

The human brain cannot make better judgments without careful observation of life. Similarly, the improved operation of mobile deep learning cannot be achieved without good contextual data input and continuous feedback. Therefore, first, we should consider how to enrich mobile information acquisition devices, which are not limited to existing cameras, microphones, temperature sensors, etc., to obtain more comprehensive feature-dimensional data. Second, how to improve the accuracy of the data acquired by mobile information acquisition devices, reduce the amount of dirty data and improve the ability to acquire accurate data in poor environments is also an area that needs improvement in mobile information collection equipment. Third, the heterogeneity of sensor quality in various devices is also one of the key issues [120]. On the one hand, mobile devices are usually equipped with nonprofessional sensors. The sensor quality of different devices may be uneven, which leads to the uneven quality of sensor data obtained. For example, in third-party mobile applications, this has a great impact on the accuracy of deep tasks. On the other hand, the workload of the mobile system is unpredictable, which may result in different sampling rates in different time periods, so the quality of sensor data may be unstable over time. Work has been done [121] to solve these problems, but there are still some shortcomings, such as the execution time and energy consumption of the whole optimization framework on mobile devices. In terms of mobile sensor data acquisition, we still need more work.

5.1.2. Mobile device design

From the current development trend of the mobile device market, it can be seen that mobile devices tend to be smaller, thinner, more portable, while reducing their size. All of these factors limit the size of the device hardware, such as the CPU, heat sink area and memory chip, which restricts the performance of MDLAs. Improving the computing and storage ability of mobile devices, similar to the introduction of deep learning chipsets and GPUs in mobile devices, is one of the key development directions of mobile devices in the future.

5.1.3. Mobile device power consumption

As is well known, the computation of most MDLAs is very powerful, especially for those vision applications. In contrast, most of the battery capacity of mobile devices is restricted and not enough to support complete local operation. Although we can offload it to the background for high-performance computing, sometimes, due to high data privacy and poor network conditions, some users may prefer to run their applications locally on the mobile end. Besides, offloading itself is also energy-

consuming work. So, how to improve the endurance of mobile devices is a key problem. We can study it by reducing the energy consumption of MDLAs and improving the battery capacity of mobile devices.

5.2. Data management

5.2.1. Data personalized management

Smart phones have become the main computing platform for millions of people. They also represent a new set of input devices, millions of cameras, microphones, GPS devices and many other types of sensors that generate massive data at every moment. With the increase in the number of connected devices, including mobile phones, tablets and laptops, there is an urgent need for personalized management of mobile user data. There are two main questions: The first is how to store massive data. First, these data cannot be completely stored in mobile devices. The limited storage capacity of a mobile device can only support it in storing a small quantity of running data and personalized digital media data. For many devices with insufficient storage, users also need to clear the cache regularly. Apple iCloud is a good example of cloud storage replacing local storage. The second point is to store a large quantity of user data in the cloud or edge servers, or even in peer devices such as domestic micro-clouds and other mobile devices. How can massive data be managed in a personalized way? For users, it is convenient for data acquisition, data consistency, data recovery, data updating and cleaning, etc. For enterprises, in core data mining, marketing, maintenance of member service data, etc., there are many factors to be considered. Because of the massive size and frequent updates of data, a more detailed design is needed in MDLA data management.

5.2.2. Data privacy and security

Offloading user data to the background will inevitably face data privacy and security problems. First, for highly private data, privacy issues in communication, such as eavesdropping, may occur in the process of offloading. Second, there is a problem of how to protect the privacy of user data when offloading the user data to the background server. Third, massive data have brought great value, including a large number of data models and information; this raises the question of how to obtain the user's consent to use them and how to protect the user's privacy while using these data with the user's consent. The former may require the design of nontechnical issues such as reward mechanisms. The latter requires the design and support of data encryption, feature abstraction and other technical issues.

5.3. System and network

5.3.1. Distributed system for MDLAs

Most offloading of MDLAs is related to the work of distributed computing and caching. The traditional design of highly available distributed systems usually needs to achieve redundancy, state synchronization, resource scheduling, system self-inspection, fault recovery, convenient scaling, etc. In the process of offloading MDLAs, we need to consider not only the typical factors above but also the characteristics of mobile devices and deep learning models, as well as the challenges brought by the diversity of offloading locations. Mobile devices have mobility and rely on unstable wireless networks and cellular connections, which makes it difficult for offloading to achieve high fault tolerance and a stable state. It also affects offloading and caching decisions by changing devices' locations among multiple servers. This raises the questions of (i) how to model the spatiotemporal characteristics of users; (ii) how to cache the user's content in a mobile-aware way; (iii) how to improve the hit rate of the edge cache when users request cached content; and (iv) how to ensure the continuity of services when the mobility of users is unknown. All of these questions have to be considered after the introduction of mobility. What's more, now we have to consider the deep learning model's distributed training and parallel inference.

5.3.2. Advancing communication

With the migration of content and computing to the edge, the vigorous development of MDLAs shows the characteristics of low delay and high reliability in computing, and distributed and high bandwidth in content. This poses the development requirements of ultra-large bandwidth, ultra-large connection, ultra-reliability and low delay for new communication networks. Bocardi's work has identified five key technologies of 5G: device-centered architecture, millimeter-wave technology, large-scale MIMO systems, more intelligent devices, and Machine-to-Machine (M2M) communication [122]. We will discuss the communication challenges of MDLAs based on these five technologies.

a) Device-centered architecture: In the past, as the basic unit of the wireless access network, the cell plays an important role in controlling the uplink and downlink transmission of data services. However, recently, the focus has gradually moved from core networks to peripheral devices, and the traditional cell-centric architecture has been destroyed. We need to redefine the network architecture of the new era, and we face some challenges. First, we study an ultra-dense heterogeneous network: MDLAs make the density of heterogeneous access in mobile cellular units rapidly increase, and the simple and single communication network architecture is not enough to meet intensive and diversified user needs. The design of communication is now affected by the type of popular MDLAs in this area and user's mobility, and the coordination compensation between different layers of the network architecture also needs to be considered; In addition, with the rapid increase of base station density, achieving more flexible adaptive resource scheduling between them for MDLAs also urgently needs to be solved. Secondly, MDLAs also need 5G technology with strong connectivity and highly intensive deployment.
b) Communication technology: (1) Millimeter waves: 5G has drawn attention to millimeter-wave, which brings greater bandwidth, richer spectrum resources, more high-frequency antennas and higher propagation accuracy. However, millimeter waves are easily affected by the environment, and the propagation distance is short, so we need more technologies to improve signal anti-interference abilities and reduce path loss. (2) Communication between mobile devices: To share content and jointly computation between mobile devices wirelessly, more efficient D2D communication need to be developed. Efforts need to be made to design user-sharing schemes, including hardware and content.

5.4. New application types

People always need more types of MDLAs and new mobile devices. To start with, the recent App market has shown us its unlimited possibilities.

For example, Apps that run on traditional mobile devices (including but not limited to recommendations from nearby smart friends) are voice recognition modules that allow users to issue voice commands in social software. Besides, the combination of visual services and DL results in a super-visual service: auto-beatifying for multi-media data, 360-degree panoramic transmission, viewpoint HD, super-resolution reconstruction, post-occlusion visual extension, visual authentication, etc. Moreover, intelligence in online shopping can be applied to building a user image for commodity recommendations, false goods, poor seller analysis [123], etc. In addition, some novel applications on new types of mobile devices have appeared in recent years, such as wearable mobile devices [124], AR/VR glasses, smart tableware [12], and driverless cars [125]. In fact, these apps are not enough. Human social activities, work activities, physical activities as well as sensory activities such as vision, hearing, smell, taste, touch and vision, hearing, smell, taste, tactile, and vision, will be combined with mobile deep learning in the future. The new design of MDLAs will make our lives much easier.

6. Conclusions

In this paper, we discuss two aspects of deploying MDLAs. One way is to execute them locally on mobile devices. The main methods include (i) reducing complexity by improving the deep learning algorithm or by redesigning the model architecture to be suitable for mobile terminals; (ii) reusing intermediate results of deep models to reduce the amount of computation; and (iii) developing a lighter deep learning framework for mobile devices. The second way is to use the cloud, mobile edge servers, and ad hoc cloudlets to enable the distributed deployment of MDLAs. We discuss this scheme from three perspectives: a distributed deployment framework, computing offloading decisions and cache configuration. In section 4, we classify current MDLAs and then give some promising future development directions for MDLAs.

Declaration of competing interest

The authors declared that they have no conflicts of interest to this work.

We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

Acknowledgments

This work is supported by the National Key Research and Development Program of China with grant number 2020AAA0108800, the National Science Foundation of China under Grant Nos. 61772414, 61532015, 61532004, 61721002, 61472317, and 61502379, the MOE Innovation Research Team No. IRT 17R86, and the Project of China Knowledge Centre for Engineering Science and Technology.

References

[1] L. Wei, W. Luo, J. Weng, Y. Zhong, X. Zhang, Z. Yan, Machine learning-based malicious application detection of android, IEEE Access 5 (2017) 25591–25601, https://doi.org/10.1109/ACCESS.2017.2771470.

[2] S. Xu, L. Zhang, A. Li, X.Y. Li, C. Ruan, W. Huang, Appdna: app behavior profiling via graph-based deep learning, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 1475–1483.

[3] L. Lu, J. Yu, Y. Chen, H. Liu, Y. Zhu, L. Kong, M. Li, Lip reading-based user authentication through acoustic sensing on smartphones, IEEE/ACM Trans. Netw. 27 (1) (2019) 1–14.

[4] B. Zhou, J. Lohokare, R. Gao, F. Ye, Echoprint:two-factor authentication using acoustics and vision on smartphones, in: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, MobiCom 2018, New Delhi, India, 2018, pp. 321–336.

[5] B. Fang, X. Zeng, M. Zhang, Nestdnn: resource-aware multi-tenant on-device deep learning for continuous mobile vision, in: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, MobiCom 2018, New Delhi, India, 2018, pp. 115–127.

[6] M. Xu, M. Zhu, Y. Liu, F.X. Lin, X. Liu, Deepcache: principled cache for mobile deep vision, in: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking(2018), MobiCom '18, ACM, New York, NY, USA, 2018, pp. 129–144.

[7] J. Ren, L. Gao, H. Wang, Z. Wang, Optimise web browsing on heterogeneous mobile platforms: a machine learning based approach, in: 2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, 2017, pp. 1–9.

[8] W. Jiang, C. Miao, F. Ma, S. Yao, Y. Wang, Y. Yuan, H. Xue, C. Song, X. Ma, D. Koutsonikolas, W. Xu, L. Su, Towards environment independent device free human activity recognition, in: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, MobiCom 2018, New Delhi, India, 2018, pp. 289–304.

[9] H. Zhang, C. Song, A. Wang, C. Xu, D. Li, W. Xu, Pdvocal: towards privacy-preserving Parkinson's disease detection using non-speech body sounds, in: The 25th Annual International Conference on Mobile Computing and Networking, MobiCom 2019, Los Cabos, Mexico, 2019, pp. 1–16.

[10] H. Zhang, C. Xu, H. Li, A.S. Rathore, C. Song, Z. Yan, D. Li, F. Lin, K. Wang, W. Xu, Pdmove: towards passive medication adherence monitoring of Parkinson's disease using smartphone-based gait assessment, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 3 (3) (2019) 123:1–123:23.

[11] H. Du, P. Li, H. Zhou, W. Gong, G. Luo, P. Yang, Wordrecorder: accurate acoustic-based handwriting recognition using deep learning, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 1448–1456. April 16-19, 2018.

[12] Q. Huang, Z. Yang, Q. Zhang, Smart-u: smart utensils know what you eat, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 1439–1447. April 16-19, 2018.

[13] T. Zhao, J. Liu, Y. Wang, H. Liu, Y. Chen, Ppg-based finger-level gesture recognition leveraging wearables, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 1457–1465. April 16-19, 2018.

[14] M. Cheung, J. She, L. Liu, Deep learning-based online counterfeit-seller detection, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2018, Honolulu, HI, USA, 2018, pp. 51–56. April 15-19,2018.

[15] Y. Zou, G. Wang, K. Wu, L.M. Ni, Smartsensing: sensing through walls with your smartphone!, in: 11th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2014, Philadelphia, PA, USA, 2014, pp. 55–63. October 28-30, 2014.

[16] T. Meng, X. Jing, Z. Yana, W. Pedrycz, A survey on machine learning for data fusion, Inf. Fusion 57 (2020) 115–129.

[17] J. Wang, X. Jing, Z. Yan, Y. Fu, W. Pedrycz, L. T. Yang, A survey on trust evaluation based on machine learning, ACM Comput. Surv. 53 (5) (2020)1-36.

[18] Z. Lu, C.K. S, P. Shiliang, L.P. Tom, Crowdvision: a computing platform for video crowdprocessing using deep learning, in: IEEE Transactions on Mobile Computing PP (99), 2018, 1–1.

[19] Y. Tian, W. Wei, Q. Li, F. Xu, S. Zhong, Mobicrowd: mobile crowdsourcing on location-based social networks, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 2726–2734.

[20] Q. Xu, R. Zheng, When data acquisition meets data analytics: a distributed active learning framework for optimal budgeted mobile crowdsensing, in: 2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, 2017, pp. 1–9.

[21] S. He, K.G. Shin, Steering crowdsourced signal map construction via bayesian compressive sensing, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 1016–1024.

[22] T. Tuor, S. Wang, T. Salonidis, B. Ko, K.K. Leung, Demo abstract: distributed machine learning at resource-limited edge nodes, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2018, Honolulu, HI, USA, 2018, pp. 1–2.

[23] S. Wang, T. Tuor, T. Salonidis, K.K. Leung, C. Makaya, T. He, K. Chan, When edge meets learning: adaptive control for resource-constrained distributed machine learning, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 63–71.

[24] Y. Bao, Y. Peng, C. Wu, Z. Li, Online job scheduling in distributed machine learning clusters, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 495–503.

[25] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282. AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA, Vol. 54 of Proceedings of Machine Learning Research.

[26] L. He, K. Ota, M. Dong, Learning iot in edge: deep learning for the internet of things with edge computing, IEEE Network 32 (1) (2018) 96–101.

[27] Y. Chen, L. Shu, L. Wang, Poster abstract: traffic flow prediction with big data: a deep learning based time series model, in: 2017 IEEE Conference on Computer Communications Workshops, INFOCOMWorkshops, Atlanta, GA, USA, 2017, pp. 1010–1011.

[28] Y. Hou, P. Zhou, J. Xu, D.O. Wu, Course recommendation of mooc with big data support: a contextual online learning approach, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, 2018, pp. 106–111.

[29] L.N. Huynh, Y. Lee, R.K. Balan, Deepmon: mobile gpu-based deep learning framework for continuous vision applications, in: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '17, ACM, New York, NY, USA, 2017, pp. 82–95.

[30] S. Han, J. Pool, J. Tran, W.J. Dally, Learning both weights and connections for efficient neural networks, in: Proceedings of the 28th International Conference on Neural Information Processing Systems, vol. 1, NIPS'15, 2015, pp. 1135–1143.

[31] S. Anwar, K. Hwang, W. Sung, Structured pruning of deep convolutional neural networks, J. Emerg. Technol. Comput. Syst. 13 (3) (2017) 32:1–32:18.

[32] F. Moya Rueda, R. Grzeszick, G.A. Fink, Neuron pruning for compressing deep networks using maxout architectures, in: V. Roth, T. Vetter (Eds.), Pattern Recognition, Springer International Publishing, Cham, 2017, pp. 177–188.

[33] W. Chen, J.T. Wilson, S. Tyree, K.Q. Weinberger, Y. Chen, Compressing neural networks with the hashing trick, Computer Science (2015) 2285–2294.

[34] Z. Li, B. Ni, W. Zhang, X. Yang, G. Wen, Performance guaranteed network acceleration via high-order residual quantization, in: 2017 IEEE International Conference on Computer Vision, ICCV), 2017, pp. 2584–2592.

[35] Z. Liu, M. Sun, T. Zhou, G. Huang, T. Darrell, Rethinking the value of network pruning, in: International Conference on Learning Representations, 2019, pp. 1–21.

[36] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, A. Krishnamurthy, Mcdnn: an approximation-based execution framework for deep stream processing under resource constraints, in: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '16, ACM, New York, NY, USA, 2016, pp. 123–136.

[37] P. Guo, B. Hu, R. Li, W. Hu, Foggycache: Cross-Device Approximate Computation Reuse, 2018, pp. 19–34, https://doi.org/10.1145/3241539.3241557.

[38] A. Mathur, N.D. Lane, S. Bhattacharya, A. Boran, C. Forlivesi, F. Kawsar, Deepeye: resource efficient local execution of multiple deep vision models using wearable commodity hardware, in: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys'17, Niagara Falls, NY, USA, 2017, pp. 68–81. June 19-23, 2017.

[39] Y. Geng, Y. Yang, G. Cao, Energy-efficient computation offloading for multicore-based mobile devices, in: INFOCOM 2018 - IEEE Conference on Computer Communications, Proceedings - IEEE INFOCOM, Institute of Electrical and Electronics Engineers Inc., United States, 2018, pp. 46–54, https://doi.org/10.1109/INFOCOM.2018.8485875.

[40] Google, Tensorflow lite. https://tensorflow.google.cn/lite/guide (accessed 31 May 2020).

[41] Facebook, caffe2. https://caffe2.ai/blog/2018/05/02/Caffe2_PyTorch_1_0.html (accessed 31 May 2020).

[42] Apple, Core ml2. https://developer.apple.com/documentation/coreml (accessed 31 May 2020).

[43] Tecent, Ncnn. https://github.com/Tencent/ncnn (accessed 31 May 2020).

[44] Tecent, Feathercnn. https://github.com/Tencent/FeatherCNN (accessed 31 May 2020).

[45] Qualcomm, Snpe, in: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk (accessed 31 May 2020).

[46] Xiaomi, Mace. https://github.com/XiaoMi/mace (accessed 31 May 2020).

[47] Amazon, Amazon ec2. https://aws.amazon.com/de/ec2/ (accessed 31 May 2020).

[48] T.Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, H. Balakrishnan, Glimpse: continuous, real-time object recognition on mobile devices, in: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, SenSys '15, ACM, New York, NY, USA, 2015, pp. 155–168.

[49] P. Jain, J. Manweiler, R. Roy Choudhury, Low bandwidth offload for mobile ar, in: Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies, CoNEXT '16, ACM, New York, NY, USA, 2016, pp. 237–251.

[50] R. Shea, A. Sun, S. Fu, J. Liu, Towards fully offloaded cloud-based ar: design, implementation and experience, in: Proceedings of the 8th ACM on Multimedia Systems Conference, MMSys'17, ACM, New York, NY, USA, 2017, pp. 321–330.

[51] Mobile-edge Computing Introductory Technical White Paper, Tech. Rep., European Telecommunications Standards Institute, 2015.

[52] S. Josilo, G. Dan, Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks, IEEE Transactions on Mobile Computing PP (99) 18 (1) (2016) 207–220.

[53] Y. Liu, M.J. Lee, Y. Zheng, Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system, IEEE Trans. Mobile Comput. 15 (10) (2016) 2398–2410.

[54] J. Zheng, C. Yueming, W. Yuan, S.X. Sherman, Dynamic computation offloading for mobile cloud computing: a stochastic game-theoretic approach, IEEE Transactions on Mobile Computing 18 (4) (2018) 771–786.

[55] J. Xu, L.C.P. Zhou, Joint service caching and task offloading for mobile edge computing in dense networks, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 207–215, https://doi.org/10.1109/INFOCOM.2018.8485977.

[56] Q. Liu, S. Huang, J. Opadere, T. Han, An edge network orchestrator for mobile augmented reality, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 756–764.

[57] M. Chen, Y. Hao, Y. Li, C.-F. Lai, D. Wu, On the computation offloading at ad hoc cloudlet: architecture and service modes, IEEE Commun. Mag. 53 (6) (2015) 18–24.

[58] Y. Mao, C. You, J. Zhang, K. Huang, K.B. Letaief, A survey on mobile edge computing: the communication perspective, IEEE Communications Surveys and Tutorials 19 (4) (2017) 2322–2358.

[59] P. Mach, Z. Becvar, Mobile edge computing: a survey on architecture and computation offloading, IEEE Communications Surveys and Tutorials 19 (3) (2017) 1628–1656.

[60] M.S. Dao, M.S. Dao, V. Mezaris, F.G.B.D. Natale, Deep learning for mobile multimedia: a survey, ACM Trans. Multimed Comput. Commun. Appl 13 (3s) (2017) 1–22.

[61] K. Kumar, J. Liu, Y.-H. Lu, B. Bhargava, A survey of computation offloading for mobile systems, Mobile Network. Appl. 18 (1) (2013) 129–140.

[62] D. Li, X. Wang, D. Kong, Deeprebirth: Accelerating Deep Neural Network Execution on Mobile Devices. in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI Press, 2018, pp. 2322–2330.

[63] S. Han, H. Mao, W. J. Dally, Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding in: International Conference on Learning Representations (ICLR), 2016,pp.1-14.

[64] P. Yin, J. Lyu, S. Zhang, S.J. Osher, Y. Qi, J. Xin, Understanding straight-through estimator in training activation quantized neural nets, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 2019.

[65] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, K. Keutzer, Squeezenet: Alexnet-Level Accuracy with 50x Fewer Parameters and <0.5mb Model Size, CoRR abs/1602.07360.

[66] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications, CoRR abs/1704.04861..

[67] C. Tai, T. Xiao, X. Wang, Convolutional neural networks with low-rank regularization, W. E, in: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2016. May 2-4, 2016, Conference Track Proceedings.

[68] J. Park, S. R. Li, W. Wen, P. T. P. Tang, H. Li, Y. Chen,P. Dubey, Faster cnns with direct sparse convolutionsand guided pruning, in: 5th International Conference onLearning Representations, ICLR, Toulon, France,April 24-26, 2017, pp.1-12

[69] C. Bucila, R. Caruana, A. Niculescu-Mizil, Model compression, in: Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 535–541.

[70] G. E. Hinton, O. Vinyals, J. Dean, Distilling the Knowledge in a Neural Network, CoRR abs/1503.02531.

[71] T. Chen, L. Lin, W. Zuo, X. Luo, L. Zhang, Learning a wavelet-like auto-encoder to accelerate deep neural networks, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence,(AAAI-18), the 30th Innovative Applications of Artificial Intelligence(IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), 2018, pp. 6722–6729.

[72] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, Q.V. Le, Mnasnet: platform-aware neural architecture search for mobile, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 2019, pp. 2820–2828.

[73] P. Zhang, E. Lo, B. Lu, High performance depthwise and pointwise convolutions on mobile devices, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI Press, 2020, pp. 6795–6802.

[74] A. Tveit, T. Morland, T. B. Røst, Deeplearningkit - an Gpu Optimized Deep Learning Framework for Apple's Ios, Os X and Tvos Developed in Metal and Swift, ArXiv abs/1605.04614.

[75] N.D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, F. Kawsar, Deepx: a software accelerator for low-power deep learning inference on mobile devices, in: 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2016, pp. 1–12, https://doi.org/10.1109/IPSN.2016.7460664.

[76] BaiDu, Paddle model. https://github.com/PaddlePaddle/models.

[77] N. Makris, V. Passas, T. Korakis, L. Tassiulas, Employing mec in the cloud-ran: an experimental analysis, in: Proceedings of the 2018 on Technologies for the Wireless Edge Workshop, EdgeTech@MobiCom 2018, New Delhi, India, 2018, pp. 15–19. November 2, 2018.

[78] W. Jiang, G. Feng, S. Qin, T.S.P. Yum, Efficient d2d content caching using multi-agent reinforcement learning, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2018, pp. 511–516.

[79] A. Mathur, N.D. Lane, S. Bhattacharya, A. Boran, C. Forlivesi, F. Kawsar, Deepeye: resource efficient local execution of multiple deep vision models using wearable commodity hardware, in: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '17, ACM, New York, NY, USA, 2017, pp. 68–81.

[80] S. Sundar, B. Liang, Offloading dependent tasks with communication delay and deadline constraint, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 37–45.

[81] H.T. Dinh, C. Lee, D. Niyato, W. Ping, A survey of mobile cloud computing: architecture, applications, and approaches, Wireless Commun. Mobile Comput. 13 (18) (2013) 1587–1611.

[82] D.V. Le, C.K. Tham, A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2018, Honolulu, HI, USA, 2018, pp. 760–765.

[83] S. Teerapittayanon, B. McDanel, H.T. Kung, Distributed deep neural networks over the cloud, the edge and end devices, in: 37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA, 2017, pp. 328–339.

[84] V.B.C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, G. Tashakor, Handling service allocation in combined fog-cloud scenarios, in: 2016 IEEE International Conference on Communications, ICC 2016, Kuala Lumpur, Malaysia, 2016, pp. 1–5.

[85] Y. Keshtkarjahromi, Y. Xing, H. Seferoglu, Dynamic heterogeneity-aware coded cooperative computation at the edge, in: 2018 IEEE 26th International Conference on Network Protocols (ICNP), 2018.

[86] X. Ge, S. Tu, G. Mao, C.-X. Wang, T. Han, 5g ultra-dense cellular networks, IEEE Wireless Communications 23 (1) (2016) 72–79.

[87] K. Poularakis, J. Llorca, A.M. Tulino, I. Taylor, L. Tassiulas, Joint service placement and request routing in multi-cell mobile edge computing networks, in: 2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, 2019, pp. 10–18.

[88] M. Chen, Y. Hao, Task offloading for mobile edge computing in software defined ultra-dense network, IEEE J. Sel. Area. Commun. 36 (3) (2018) 587–597.

[89] F. Zafari, J. Li, K. K. Leung, D. Towsley, A. Swami, A Game-Theoretic Approach to Multi-Objective Resource Sharing and Allocation in Mobile Edge Clouds, CoRR abs/1808.06937.

[90] X. Ran, H. Chen, X. Zhu, Z. Liu, J. Chen, Deepdecision, A mobile deep learning framework for edge video analytics, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 1421–1429, https://doi.org/10.1109/INFOCOM.2018.8485905.

[91] X. Ran, H. Chen, Z. Liu, J. Chen, Delivering deep learning to mobile devices via offloading, in: Proceedings of the Workshop on Virtual Reality and Augmented Reality Network, VR/AR Network'17, ACM, New York, NY, USA, 2017, pp. 42–47.

[92] J. Liu, Y. Mao, J. Zhang, K.B. Letaief, Delay-optimal computation task scheduling for mobile-edge computing systems, in: 2016 IEEE International Symposium on Information Theory, ISIT), 2016, pp. 1451–1455, https://doi.org/10.1109/ISIT.2016.7541539.

[93] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, K. Ha, K. Elgazzar, P. Pillai, R. Klatzky, D. Siewiorek, M. Satyanarayanan, An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, SEC '17, ACM, New York, NY, USA, 2017, pp. 14:1–14:14.
[94] M. Kamoun, W. Labidi, M. Sarkiss, Joint resource allocation and offloading strategies in cloud enabled cellular networks, in: 2015 IEEE International Conference on Communications, ICC), 2015, pp. 5529–5534, https://doi.org/10.1109/ICC.2015.7249203.
[95] W. Labidi, M. Sarkiss, M. Kamoun, Energy-optimal resource scheduling and computation offloading in small cell networks, in: 2015 22nd International Conference on Telecommunications, ICT), 2015, pp. 313–318, https://doi.org/10.1109/ICT.2015.7124703.
[96] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM Trans. Netw. 24 (5) (2016) 2795–2808, https://doi.org/10.1109/TNET.2015.2487344.
[97] H. Flores, P. Hui, P. Nurmi, E. Lagerspetz, S. Tarkoma, J. Manner, V. Kostakos, Y. Li, X. Su, Evidence-aware mobile computational offloading, IEEE Trans. Mobile Comput. 17 (8) (2018) 1834–1850, https://doi.org/10.1109/TMC.2017.2777491.
[98] W.R. KhudaBukhsh, B. Alt, S. Kar, A. Rizk, H. Koeppl, Collaborative uploading in heterogeneous networks: optimal and adaptive strategies, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 1–9.
[99] X. Wang, M. Chen, T. Taleb, A. Ksentini, V.C.M. Leung, Cache in the air: exploiting content caching and delivery techniques for 5g systems, IEEE Commun. Mag. 52 (2) (2014) 131–139.
[100] L. Wang, L. Jiao, T. He, J. Li, M. Mühlhäuser, Service entity placement for social virtual reality applications in edge computing, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, 2018, pp. 468–476.
[101] J. Su, D.V. Vargas, K. Sakurai, One Pixel Attack for Fooling Deep Neural Networks, IEEE Transactions on Evolutionary Computation 23 (5) (2019) 828–841.
[102] K.-H.K. Xiufeng Xie, Source compression with bounded dnn perception loss for iot edge computer vision, in: Proceedings of the 25th Annual International Conference on Mobile Computing and Networking(2019), MobiCom '19, ACM, 2019, pp. 1–16.
[103] Y. Zhang, X. Jiang, Y. Wang, K. Lei, Cache and delivery of vr video over named data networking, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2018, Honolulu, HI, USA, 2018, pp. 280–285.
[104] H. Hao, C. Xu, M. Wang, H. Xie, Y. Liu, D.O. Wu, Knowledge-centric proactive edge caching over mobile content distribution network, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2018, Honolulu, HI, USA, 2018, pp. 450–455.
[105] N. Mohan, P. Zhou, K. Govindaraj, J. Kangasharju, Managing data in computational edge clouds, in: Proceedings of the Workshop on Mobile Edge Communications, MECOMM@SIGCOMM 2017, Los Angeles, CA, USA, 2017, pp. 19–24.
[106] J. Gu, W. Wang, A. Huang, H. Shan, Proactive storage at caching-enable base stations in cellular networks, in: 24th IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications, PIMRC 2013, London, United Kingdom, 2013, pp. 1543–1547.
[107] F. Wang, F. Wang, J. Liu, R. Shea, L. Sun, Intelligent video caching at network edge: a multi-agent deep reinforcement learning approach, in: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 2020, pp. 2499–2508.
[108] A. Raman, N. Sastry, A. Sathiaseelan, J. Chandaria, A. Secker, Wi-stitch: content delivery in converged edge networks, in: Proceedings of the Workshop on Mobile Edge Communications, MECOMM@SIGCOMM 2017, Los Angeles, CA, USA, 2017, pp. 13–18.
[109] A. Raman, N. Sastry, N. Mokari, M. Salehi, T. Faisal, A. Secker, J. Chandaria, Care to share?: an empirical analysis of capacity enhancement by sharing at the edge, in: Proceedings of the 2018 on Technologies for the Wireless Edge Workshop, EdgeTech@MobiCom 2018, New Delhi, India, 2018, pp. 27–31.
[110] A. Mete, S. Moharir, Caching policies for d2d-assisted content delivery systems, in: Proceedings of the 2018 on Technologies for the Wireless Edge Workshop, ACM, 2018, pp. 3–7.
[111] X. Zhang, Q. Zhu, D2d offloading for statistical qos provisionings over 5g multimedia mobile wireless networks, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 82–90.
[112] D. Liu, B. Chen, C. Yang, A.F. Molisch, Caching at the wireless edge: design aspects, challenges, and future directions, IEEE Commun. Mag. 54 (9) (2016) 22–28.
[113] A. Tatar, M.D.D. Amorim, S. Fdida, P. Antoniadis, A survey on predicting the popularity of web content, Journal of Internet Services and Applications 5 (1) (2014) 8, https://doi.org/10.1186/s13174-014-0008-y.
[114] C. Bernardini, T. Silverston, F. Olivier, Mpc:popularity-based caching strategy for content centric networks, in: Proceedings of IEEE International Conference on Communications, ICC 2013, Budapest, Hungary, 2013, pp. 3619–3623.
[115] J. Li, S. Shakkottai, J.C.S. Lui, V. Subramanian, Accurate learning or fast mixing? dynamic adaptability of caching algorithms, IEEE J. Sel. Area. Commun. 36 (6) (2018) 1314–1330.
[116] S. Misra, N. Saha, Detour: dynamic task offloading in software-defined fog for iot applications, IEEE J. Sel. Area. Commun. 37 (5) (2019), 1–1.
[117] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, L. Tassiulas, Joint Service Placement and Request Routing in Multi-Cell Mobile Edge Computing Networks, CoRR abs/1901.08946.
[118] H. Gong, K. Xing, W. Du, A user activity pattern mining system based on human activity recognition and location service, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2018, Honolulu, HI, USA, 2018, pp. 1–2.
[119] H. Zhang, A. Wang, D. Li, W. Xu, Deepvoice: a voiceprint-based mobile health framework for Parkinson's disease identification, in: 2018 IEEE EMBS International Conference on Biomedical & Health Informatics, BHI 2018, Las Vegas, NV, USA, 2018, pp. 214–217.
[120] A. Stisen, H. Blunck, S. Bhattacharya, T.S. Prentow, M.B. Kjærgaard, A.K. Dey, T. Sonne, M.M. Jensen, Smart devices are different: assessing and mitigatingmobile sensing heterogeneities for activity recognition, in: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, SenSys 2015, Seoul, South Korea, 2015, pp. 127–140.
[121] S. Yao, Y. Zhao, H. Shao, D. Liu, S. Liu, Y. Hao, A. Piao, S. Hu, S. Lu, T.F. Abdelzaher, Sadeepsense: self-attention deep learning framework for heterogeneous on-device sensors in internet of things applications, in: 2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, 2019, pp. 1243–1251.
[122] F. Boccardi, R.W. Heath, A. Lozano, T.L. Marzetta, P. Popovski, Five disruptive technology directions for 5g, IEEE Commun. Mag. 52 (2) (2014) 74–80.
[123] M. Cheung, J. She, L. Liu, Deep learning-based online counterfeit-seller detection, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2018, Honolulu, HI, USA, 2018, pp. 51–56.
[124] W. Chang, Y. Yu, J. Chen, Z. Zhang, S. Ko, T. Yang, C. Hsu, L. Chen, M. Chen, A deep learning based wearable medicines recognition system for visually impaired people, in: IEEE International Conference on Artificial Intelligence Circuits and Systems, AICAS 2019, Hsinchu, Taiwan, 2019, pp. 207–208.
[125] C. Hodges, S. An, H. Rahmani, M. Bennamoun, Deep learning for driverless vehicles, in: Handbook of Deep Learning Applications, 2019, pp. 83–99.