A Survey of Deep Learning on Mobile Devices: Applications, Optimizations, Challenges, and Research Opportunities

This article surveys the state-of-the-art literature that focuses on deep learning optimization for mobile applications and devices.

By Tianming Zhao, *Member IEEE*, Yucheng Xie, Yan Wang, *Member IEEE*, Jerry Cheng, Xiaonan Guo, *Member IEEE*, Bin Hu, and Yingying Chen, *Fellow IEEE*

ABSTRACT | Deep learning (DL) has demonstrated great performance in various applications on powerful computers and servers. Recently, with the advancement of more powerful mobile devices (e.g., smartphones and touch pads), researchers are seeking DL solutions that could be deployed on mobile devices. Compared to traditional DL solutions using cloud servers, deploying DL on mobile devices have unique advantages in data privacy, communication overhead, and system cost. This article provides a comprehensive survey for the current studies of adopting and deploying DL on mobile devices. Specifically, we summarize and compare the state-of-the-art DL techniques on mobile devices in various application domains involving vision, speech/speaker recognition, human activity recognition, transportation mode detection, and security. We generalize an optimization pipeline for bringing DL to mobile devices, including model-oriented optimization mechanisms (e.g., pruning and quantization) and nonmodel-oriented optimization mechanisms (e.g., software accelerator and hardware design). Moreover, we summarize popular DL libraries regarding their support to state-of-the-art models (software) and processors (hardware). Based on our summarization, we further provide insights into potential research opportunities for developing DL for mobile devices.

KEYWORDS | Deep learning (DL); hardware and software accelerator design; mobile security; mobile sensing; optimization.

Manuscript received November 13, 2021; revised February 1, 2022; accepted February 15, 2022. Date of current version March 9, 2022. This work was supported in part by the National Science Foundation under Grant CCF2000480, Grant CCF2028858, Grant CCF2028894, Grant CCF2028873, Grant CCF2028876, Grant CCF1909963, Grant CNS1815908, and Grant CNS1717356. *(Corresponding author: Tianming Zhao.)*

Tianming Zhao and **Yan Wang** are with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122 USA (e-mail: tum94362@temple.edu; y.wang@temple.edu).
Yucheng Xie is with the Department of Electrical and Computer Engineering, Indiana University–Purdue University Indianapolis, Indianapolis, IN 46202 USA (e-mail: yx11@iupui.edu).
Jerry Cheng is with the Department of Computer Science, New York Institute of Technology, New York, NY 10023 USA (e-mail: jcheng18@nyit.edu).
Xiaonan Guo is with the Department of Computer and Information Technology Indiana University–Purdue University Indianapolis, Indianapolis, IN 46202 USA (e-mail: xg6@iupui.edu).
Bin Hu and **Yingying Chen** are with the Department of Electrical and Computer Engineering, Rutgers University, New Brunswick, NJ 08901 USA (e-mail: bh439@scarletmail.rutgers.edu; yingche@scarletmail.rutgers.edu).

Digital Object Identifier 10.1109/JPROC.2022.3153408

I. INTRODUCTION

Recent developments of deep learning (DL) have made huge leaps in various mobile applications, such as image recognition [1]–[7], speech recognition [8]–[11], and human activity recognition [12]–[17]. In general, DL could provide much higher inference accuracy than traditional machine learning methods. However, DL usually involves complicated neural network models and possibly millions of parameters that require high computational power and large memories, which are not available on mobile devices. As a result, many mobile applications choose to use cloud computing technology to offload all or partial computational tasks of DL from resource-constrained mobile devices. While the cloud-based approaches are common and have initial success in many applications, people have privacy concerns because sensitive data need to be uploaded to cloud servers. Besides, the latency of cloud-based approaches is highly affected by network conditions and the interfaces (e.g., long-term evolution (LTE) and Wi-Fi) [18]. In addition, due to high maintenance costs, cloud computing services are primarily provided by a limited number of large companies that have the resources to support large systems [19]. Therefore, cloud-based approaches can no longer satisfy the emerging artificial intelligence (AI) and augmented reality (AR) mobile applications, and new technologies aiming at enabling DL on mobile hardware are much demanded.

Many research works have been conducted to develop DL-based approaches that are optimized for mobile devices. Topics of them span over a broad range, including miniature DL model design, DL model optimization, software accelerator, and hardware accelerator. A couple of earlier studies of DL have been conducted for mobile device applications. Nan *et al.* [57] explored typical DL compression methods and compared their practical performances using various metrics. Based on their performances, they discussed the advantages and bottlenecks of those compression methods. Solutions to reduce the computational cost of DL with respect to algorithmic design, computational optimization, and hardware revolution have been explored in a more recent survey [58]. Moreover, some survey papers [59], [60] compare various attributes (e.g., latency, energy efficiency, costs, and scalability) of different mobile devices using DL and propose several evaluation criteria to choose appropriate development frameworks. In addition, Zhang *et al.* [61] presented a survey of mobile and wireless networking research based on DL. They reviewed recent DL applications to mobile and wireless networking and discussed how to tailor DL models for those problems. Chen and Ran [62] summarized DL in edge computing with optimization methods for fast inference on resource-constrained edge devices. Deng *et al.* [63] provided more optimal solutions to key problems in edge computing with the help of DL technologies and study how to carry out the entire process of building DL models on the edge. Chen and Ran [62] provided an accessible introduction to various aspects on the recent advancements of DL on mobile devices, including mobile hardware architectures, optimization approaches, existing resources, mobile sensing applications, and challenges. While these works have looked into various aspects in DL optimized for edge and mobile devices, they either mainly discuss the DL optimization (e.g., model parallelism) on edge devices for applications such as mobile and wireless networks or provide the introduction-level survey that is dedicated for quickly grasping the concepts. However, a comprehensive study of the technologies that could enable DL on various types of mobile hardware is still missing. We propose an optimization pipeline that is designed to adaptively support DL deployment on various kinds of mobile devices for different application domains. This will provide in-depth guidance for the researchers to customize their DL-based applications' deployment.

This article extensively studies the state-of-the-art literature that focuses on DL optimization for mobile applications and devices. We summarize DL-based approaches in different mobile applications in terms of evaluation platforms, adopted models, and optimization methods. In addition, we present the challenges of applying DL on mobile devices and introduce an optimization pipeline for guiding the optimization of DL targeting mobile devices and applications. Specifically, we divide the optimization pipeline into two parts, i.e., model-oriented and nonmodel-oriented, which cover typical optimization approaches with respect to both hardware and software. Moreover, we summarize software and libraries used in DL in terms of software and hardware optimization support, which could help us choose appropriate software and hardware combinations for their application development. We also point out the current trend and potential direction of research on DL optimization for various mobile applications and resource-constrained mobile devices. It is worth noting that the mobile devices mentioned in this survey include a wide range of devices from resource-constrained Internet of Things (IoT) devices to high-end smartphones.

The remaining of this article is organized as follows. Section II summarizes DL in different mobile applications. Section III discusses the challenges of applying DL on mobile devices and summarizes the optimization approaches. Section IV compares the state-of-the-art DL libraries. Section V discusses the on-device optimization in federated learning (FL). Section VI points out the potential research in different mobile applications. Section VII concludes this article.

II. DL-ENABLED MOBILE APPLICATIONS

Many mobile applications adopt DL-based approaches due to their better performance than traditional machine learning-based approaches. In this section, we review the DL approaches developed for several popular application domains, including image classification and object detection, speech and speaker recognition, activity recognition, and mobile security. Our summary mainly focuses on the following aspects: the adopted evaluation platform, the adopted DL model, and the adopted performance improvement method.

A. DL in Image Classification and Object Detection

In the past few years, decent performance has been achieved in image classification and object detection using DL approaches on mobile platforms (e.g., smartphones [1]–[7]; embedded systems including NVIDIA Jetson TK1, Intel Edison platform, and Raspberry pi [6], [39]–[41], [43], [67]–[69]; and the mobile development kit such as Snapdragon 820 and Snapdragon 845 [43]–[45]). We summarize the popular DL models for image classification and object detection in Table 1. In particular, VGG, MobileNet V1 and V2, AlexNet, LeNet, SqueezNet, GoogleNet, and ShuffleNet are usually trained with popular database (e.g., ImageNet 2012 [1], [3]–[6], [40], [44], [70], [71] and MNIST [39], [40], [70]). For object detection, popular DL models, such as SSD300, SSD512, YOLOv2, YOLOv3, YOLOv3-Tiny, SSD+MobileNet, and SSDlite+MobileNet v2, are trained

Table 1 Image Classification and Speech Recognition Applications on Mobile Devices

Application	**Image Classification/Object Detection**		**Speech/Speaker Recognition**	
	Facial expression recognition, Road damage/Crack detection, etc. [1]–[7], [20]–[45]		Speech enhancement, Voice activity/Noise detection, etc. [8]–[11], [46]–[56]	
Adopted Model	**Types**	**Complexity (architecture/layers)**	**Types**	**Complexity (architecture/layers)**
	CNN (Image Classification)	GoogLeNet [20], [24], [34]; ResNet50 [22], [32], [36]; MobileNet V1, V2 [36]; DarkNet-19 [37]; VGG 16,19 [32]; AlexNet [20], [38]; InceptionV3 [32], [36]; CNN 5 [26], 7 [25], 8 [30], 9 [27], 12 [28] layers	CNN	SqueezeNet V1.1 [56], TrimNet [8]; CNN 2 [56], 3 [56], 5 [46], 8 [47], 15 [52], 21 [51] layers
			DNN	3 [55], 5 [49], 6 [55], 8 [9]
	CNN (Object Detection)	YOLO v2 [23], [29], [37]; SSD [29], [33]; NIN [24]; Fast-RCNN [21], [29]; AlexNet [24]; InceptionV3 [22]; VGG 19 [22]	RNN(LSTM)	3 [54], 5 [53], 8 [10], 17 [11]
			AE	5 [48]
Adopted DL Library	Caffe [3], [4], [20], [24], [30], [31], [38], [40] Tensorflow [2], [33], [35] Keras [32] Theano [25], [39] Torch [41]		Tensorflow [46], [47], [50] Keras [51] Theano [49]	
Adopted Optimization	Compression (pruning) [1], [3], [39], [40]		Compression(lower rank matrices) [10], [53], [54]	
	Quantization [4], [40], [43]		Quantization [9], [11]	
	Model scaling [5], [7], [44]		Programming strategies [9]	
	Programming strategies [4]			

with the databases, such as COCO [5], [6], [45], VOC [6], [43], [45], and Cifar [2], [39]. DL approaches have also been adopted in a range of special applications, such as computed tomography (CT) image processing [72] and water ecosystem images captured by the IoT devices [73]. Among all these models, some DL models (e.g., MobileNet in Fig. 1 and SSD) are the most popular and usually adopted as the baseline for performance comparison with the newly proposed models (e.g., PeeletNet, Enet, and ThunderNet) [2], [4]–[6], [41], [42], [44], [45], [68], [71].

Recently, new DL models are becoming faster, smaller, and more power-efficient. For example, Mnasnet [5] adopted an automated mobile neural architecture search (MNAS) approach to achieve 1.8 times faster than MobileNetV2, PeleeNet [6] was 1.8 times faster than MobileNetV2 by using a dynamic number of channels in bottleneck layers, and Shufflenet [71] achieved 13 times faster than AlexNet by utilizing two new operations, pointwise group convolution and channel shuffle, to greatly reduce computation cost. Moreover, Mobiface [42] adopted two design strategies, bottleneck residual block with expansion layers and fast downsampling, to have its model size ten times smaller than MobilenetV1. However, most existing DL models still have big sizes and complicated structures, making it hard to perform real-time image classification and object detection on resource-constrained mobile devices. To this end, compression and quantization methods have been adopted to reduce the complexity and improve the inference speed

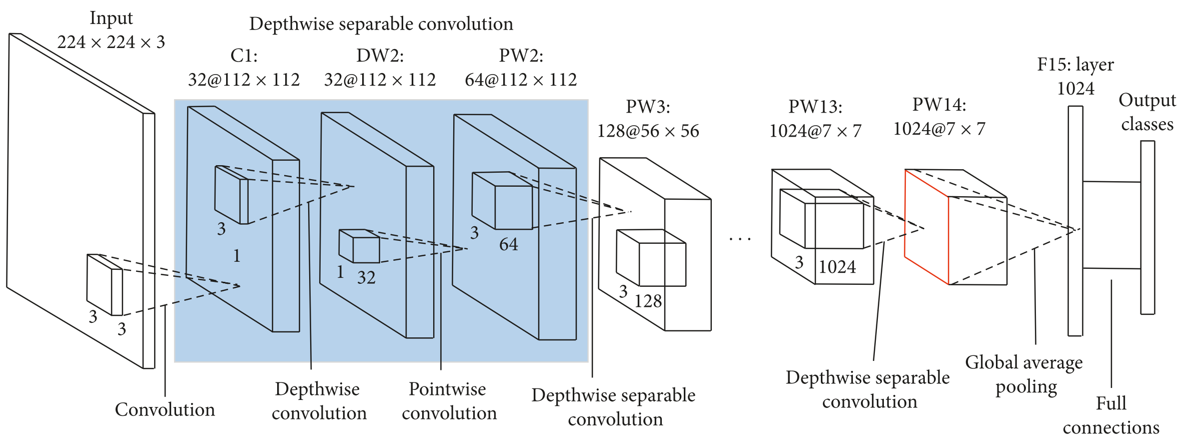


Fig. 1. ***Structure of MobileNet is based on depthwise separable filters for image classification [64]. MobileNet is a streamlined architecture that uses depthwise separable convolutions to construct lightweight deep CNNs and provides an efficient model for mobile and embedded vision applications. The model is composed of standard convolution layer (C), depthwise convolution layers (DW), pointwise convolution layers (PW), and full connection layers (F). The model also adopts a novel structure called depthwise separable convolution layer, which is a depthwise convolution layer followed by a pointwise convolution layer. Such a design could significantly reduce the model's parameters compared to the models composed of regular convolutions with the same depth, making MobileNet nearly as accurate as VGG16 [65] but 32 times smaller sized and 27 times less compute-intensive. Moreover, MobileNet can achieve 4% better accuracy than AlexNet [66] with a 45 times smaller size and 9.4 times less computation.***

Table 2 Activity Recognition and Mobile Security Applications on Mobile Devices

Application	**Activity Recognition**		**Mobile Security**	
	Human Activity Recogniton [12]–[15], [74]–[94] Transportation Mode Detection [95]–[100]		Malware Detection [101]–[107] Traffic Classification [108]	
Adopted Model	**Types**	**Complexity (layers)**	**Types**	**Complexity (layers)**
	CNN	4 [76], 5 [90], 6 [74], [79], [81], 7 [88], 8 [75], 10 [77], 13 [96], [97], 16 [85]	CNN	4 [102], 7 [103], [104], [108]
	DNN	5 [99], 10 [98]	DNN	3 [99]
	MLP	5 [83], 10 [84]	MLP	1 [108], 3 [101]
	SAE	4 [78]	SAE	5 [108]
	DBN	4 [76]	ANN	3 [106]
	RNN(LSTM)	6, 7 [80]	RNN(LSTM)	5 [108]
Adopted DL Library	Tensorflow [75], [79], [89], [96], [97], [100] Keras [77], [84] Matlab tool [78], [85], [86] Torch [82] Pylearn [89] H2O (R) [92]		Tensorflow [102], [104], [108]	
Adopted Optimization	Compression [109] Quantization [13]		Quantization [104]	

of DL models. In particular, AutoML [1] can reduce the model size of MobileNet by 34% from leveraging reinforcement learning to provide the model compression policy. DeepRebirth [3] achieved more than 3 times faster speed and 2.5 times run-time memory saving on GoogLeNet by the compression technique containing two operations (i.e., streamline slimming and branch slimming). DeepIoT [39] compressed neural network structures into smaller dense matrices by finding the minimum number of nonredundant hidden elements. It is able to reduce the model size of VGGNet by 97.6%, running time by 94.5%, and energy consumption by 95.6%. Deep compression [40] pruned unimportant connections, quantized the network using weight sharing, and then applied the Huffman coding on AlexNet, VGG-16, and LeNet to reduce their weight storage by 35, 49, and 39 times, respectively. Along this line, model scaling methods are adopted by some works [5], [7], [44], which use the depth multiplier to change the width of channel of each layer and reduce the inference time. Moreover, advanced programming strategies [e.g., fused tile partitioning (FTP)] were adopted by Zhao *et al.* [110] to reduce the memory footprint of YOLOv2 by more than 68% without sacrificing accuracy.

Although these methods can improve DL performance on mobile devices, some works, such as GoogLeNet and Thunder [45], still cannot provide real-time performance on mobile devices. The compressed GoogLeNet model has been shown good accuracy on low-end mobile devices (e.g., Moto E) [3], [4], but they can only process video streams at 10 frames/s. Thunder with SNet535 targeted at achieving high accuracy but can only process 5.8 frames/s, which is not fast enough in real time. Therefore, achieving both high accuracy and near real-time inference using DL models on mobile devices is still a challenging research problem.

B. DL in Speech and Speaker Recognition

In the speech/speaker recognition area, most existing works [51], [53]–[56], [111] claim that their proposed systems have the potential for mobile devices due to their less resource consumption. However, only a few of them [8]–[11] have demonstrated their practicality in mobile applications. We summarize the models, software platform, and optimization methods of the existing work in speech and speaker recognition, as shown in Table 1.

In the speech/speaker recognition area, recurrent neural network (RNN) architectures are commonly used because they allow extracting past and future dependencies at a given point of the audio. Some works [9], [11], [51], [52], [55] propose simple lightweight neural networks (e.g., around five-layer convolutional neural network (CNN), deep neural networks (DNN), and RNN) as shown in Fig. 2 and compare with classic baseline models. For example, He *et al.* [11] proposed a compact end-to-end (E2E) speech recognizer, which improved the word error rate by more than 20% over a conventional connectionist temporal classification (CTC)-based baseline system; Deep KWS [55] achieved 45% relative improvement in false reject rate with respect to a competitive baseline hidden Markov model (HMM). These initial results reveal the advantages and potential of developing novel DL models for speech and speaker recognition. In addition, we observe that most of the existing studies in speech recognition are evaluated against baseline models, comprised of some classical machine learning models. In the future, DL models can be used as baseline standards.

Several methods have been adopted to improve the performance in speech/speaker recognition, including compression (e.g., lower rank matrices) [10], [53], [54], quantization (e.g., floating point to 8 bit) [9], [11], lightweight neural network models (e.g., around five-layer CNN, DNN, and RNN) [9], [10], [52]–[56], and programming strategies (i.e., multithreads and frame skipping) [9]. After adopting those methods, the proposed DL model size is reduced several times compared with the original model, and the inference time is also reduced significantly. For example, the model size reduces to 1/3 times [9], [53] or 1/4 times [11], [54] of the original size, the inference time is claimed to be many times faster than the original model [10], [53], [55], [56], and the power consumption is reduced to an acceptable level [111]. However, since most of these DL models have not been deployed in

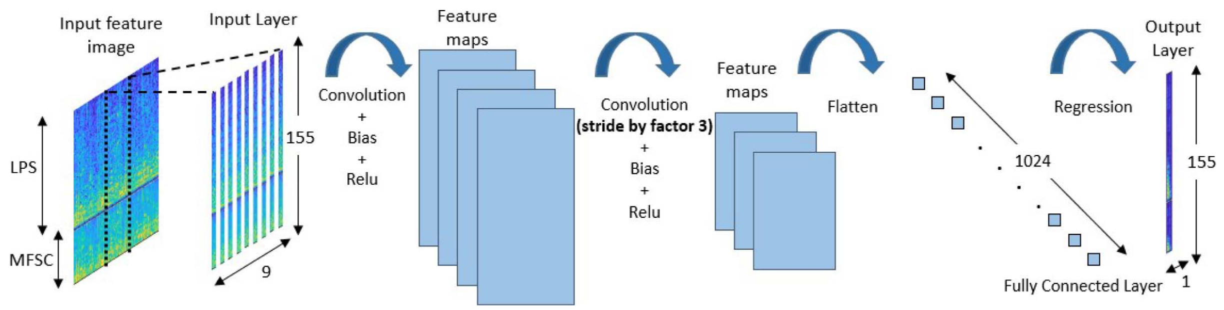


Fig. 2. *CNN-based speech/speaker recognition DL model example [47]. The architecture has three hidden layers, two convolutional layers, and one fully connected layer. The audio input is represented as short-time Fourier transform (STFT) spectrograms and is fed as an input to the CNN for audio classification tasks.*

mobile platforms and their performance improvements are only compared with the original DL models, it is hard to guarantee that these models are suitable for mobile devices. Therefore, future research studies are needed to fully optimize the DL models to satisfy the hardware resource constraint and real-time application nature in speech or speaker recognition on mobile devices.

C. DL in Activity Recognition

Activity recognition is another active research area in mobile computing, which could facilitate many emerging applications, including AI and AR/virtual reality (VR) on mobile devices. Existing researchers have adopted DL in activity recognition applications (e.g., human activity recognition and transportation mode detection) on smartphones [12]–[17], and embedded devices (e.g., Raspberry PI and MinnowBoard Turbot Dual-Core Board) [93], [94], [109]. We reviewed the recent literature on activity recognition and summarize them in Table 2 based on their complexity, software platform, and optimization methods.

Unlike the image classification area that always adopts complex DL models, the activity recognition domain usually adopts customized lightweight DL models, as shown in Fig. 3. For example, some of them adopt CNN, DNN, long short-term memory (LSTM), and so on with less than ten layers, as shown in Table 2. Although these models achieve good performance, their model structures and databases are mostly not publicly available, and thus, it is hard to compare the performance among these approaches. Therefore, open-sourced DL models and datasets are highly demanded.

We also observe that the majority of DL-based activity recognition systems are developed and evaluated based on personal computers without considering the limited resources in mobile devices. Only a few researchers have explored to improve the performance of DL models on mobile devices. As shown in Table 2, Zebin *et al.* [13] used quantization in their method, and Stardust [109] compressed the structure of DL models to fit them into mobile devices. Some works [12]–[14], [93] adopted lightweight DL models (e.g., three-, four-, and eight-layer CNNs and two-layer LSTM) to satisfy the requirements of their proposed applications. Since some lightweight DL models already achieved good performances (e.g., over 95% accuracy in five-class activity recognition [13]), further optimization is not vital to them. However, for complex activity recognition applications, DL models usually have complicated and huge-sized structures that cannot be miniaturized by using a simple method. Thus, new research on how to use the combination of multiple optimization methods (e.g., factorization and pruning) to reduce the size of such complex DL models while achieving high accuracy is essential for deploying complex DL models on mobile devices.

D. DL in Mobile Security

With the ever-increasing use of mobile and IoT devices, security is becoming more important these days. There-

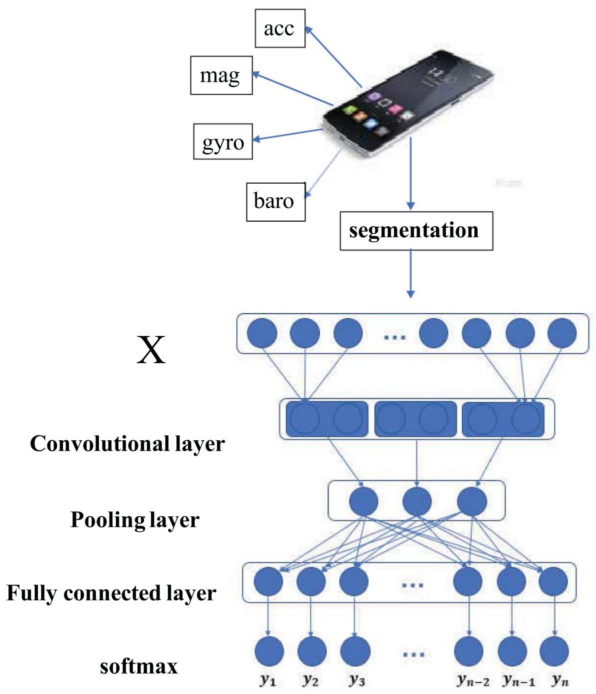


Fig. 3. *CNN-based activity recognition DL model example [90]. The network consists of three types of layers, including convolutional layer, pooling layer, and fully connected layer. The input data are a vector containing data collected from accelerometer (acc), magnetometer (mag), gyroscope (gyro), and barometer (baro).*

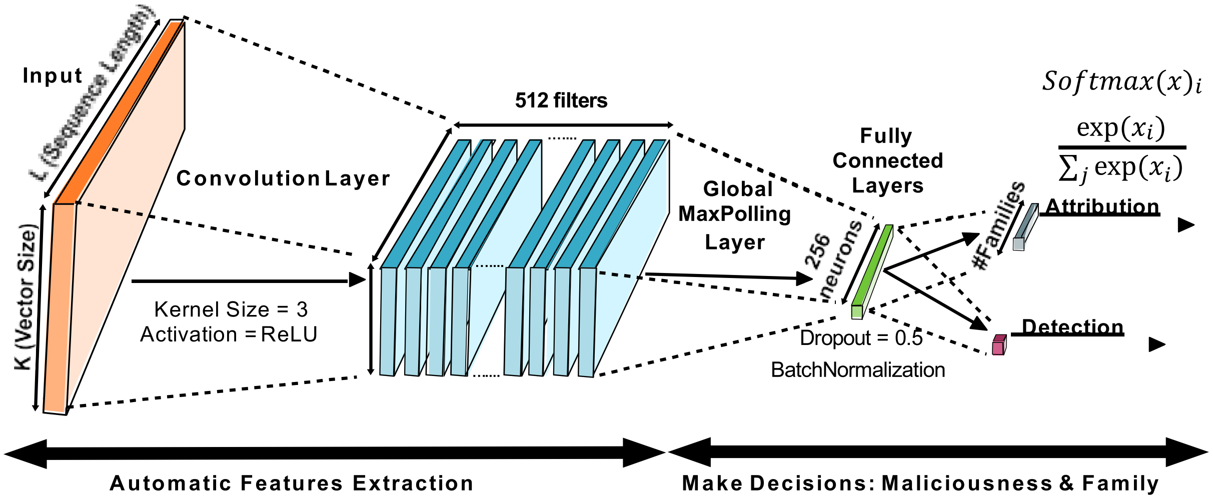


Fig. 4. ***Security DL model example [102]. The model contains a convolution layer followed by a global max pool layer and then connects to a fully connected layer. In addition to dropout used to prevent overfitting, they also utilize batch normalization to improve the results.***

fore, DL models for mobile security are also evolving quickly with recent development [101], [102], [104] for different resource constraints. Some of them are simulated using the field-programmable gate array (FPGA) emulator (e.g., Xilinx Zynq-7000) [106] and Android emulator [105]. The existing works on DL models for mobile security are summarized in Table 2. The lightweight DL models are often deployed on mobile devices, as shown in Fig. 4. For example, Yuan *et al.* [101] designed a lightweight on-device Android malware detector that is a three-layer multilayer perceptron (MLP) neural network mainly using one-shot computation for model training. Hence, it can be fully or incrementally trained directly on mobile devices (e.g., Samsung Galaxy S9). MalDozer [102] detected Android malware by using a four-layer CNN that is not only deployed on servers but also on the IoT devices (e.g., Raspberry Pi 2). Different from the aforementioned existing works that need the hand-engineered malware features, McLaughlin *et al.* [103] developed a five-layer CNN-based Android malware detection system that is capable of simultaneously learning to perform feature extraction and malware classification. Even though those models provide decent accuracy on mobile devices, they are not generally applicable to various mobile security applications since they are only trained and tested on a few malware datasets (e.g., VirusShare and Contagio, Malgenome and Drebin, and Android malware repository from McAfee Labs are adopted in [101]–[103], respectively). Also, the proposed models are mostly not available. Considering the varying nature of malware in real scenarios, public malware datasets and open-sourced DL models are desirable in mobile security.

In addition, DL is applied as a viable strategy for network traffic monitoring and analysis applications, e.g., traffic classification and prediction [112]–[116], because a large portion of malware is spreading through the network traffic and hiding activities using various encryption protocols. However, the current state of research on mobile traffic classification has not reached a similar level of DL in other fields [108]. Research about mobile networking and traffic monitoring has mostly been studied independently. Recently, only a few crossover studies [107], [108] between the two research areas have emerged. Aceto *et al.* [108] presented a review for mobile traffic classification works that use DL techniques. They reproduced several DL classifiers, e.g., MLP, LSTM, CNN, and SAE, from the traffic classification literature and also made a detailed evaluation to compare the accuracy of these classifiers. Moreover, they found that the key issues of DL in traffic classification are the lack of advanced hybrid DL architectures and the lack of up-to-date human-generated public datasets. This problem is further worsened in mobile scenarios because the possibility of sharing large and up-to-date datasets is restricted by both higher privacy concerns and the fast-paced evolution of mobile traffic mix.

Because a timely response to malicious attacks could help prevent users' loss, mobile security applications are susceptible to inference time. Usually, it requires reducing the model size and computational cost of DL models to achieve real-time detection. However, we find that the majority of existing works mainly focused on improving inference accuracy using offline computing. While high accuracy has been achieved, not much work has been done to reduce the model size and computational cost of DL models in mobile security. To achieve the real-time nature of the security applications on various mobile devices with different resource constraints, researchers will need to adaptively consider some typical performance improvement methods (e.g., quantization and pruning) in their research. Especially, when more generally applicable models with deeper architecture are adopted, more optimization methods would be needed to achieve real time on the mobile devices.

III. TOWARD ENABLING DL ON MOBILE DEVICES

A. Current Status and Issues

Currently, most of the popular DL-based mobile applications collect sensor data from mobile devices and perform inferences using DL on powerful cloud servers. This is mainly because state-of-the-art DL models usually have complex and bulky structures, which contain hundreds, even thousands of nodes, and potentially millions of parameters that cannot be deployed on resource-limited mobile devices. We have reviewed a broad range of literature about enabling DL on mobile devices and summarize current status and issues in this area as follows.

1) How to Select/Design a Suitable Model for Heterogeneous Mobile Devices and Applications Is Not Systematically Studied?: We have seen that many application domains adopt various DL models [1]–[17] on server or desktop; however, there are no general models designed specifically for different domains on mobile devices. As the first step toward enabling DL on mobile devices, the current mobile computing domain needs the capability to select and design DL models suitable for various mobile devices and applications.

2) What Kind of DL Optimization Approaches That Could Enable Existing DL Models on Various Mobile Devices Is Still Lacking Comprehensive Analysis and Research?: While many researchers have been working on DL optimization in recent years, most of them focus on reducing the model size on a desktop or server that usually has sufficient hardware resources to run various compressed models smoothly. Differently, due to various hardware resource limitations [117]–[120] on a range of mobile devices, deploying DL models on mobile devices need to consider more aspects (e.g., storage space, inferencing speed, power consumption, and real-time memory usage). No single optimization approach (e.g., weight pruning [121]) can easily meet the requirements for all those aspects. It is critical to find an optimization pipeline that adaptively adopts various exiting optimization approaches for DL deployment on mobile devices considering different application domains.

3) How to Utilize the Full Power of Existing Mobile Device Hardware and Design New Dedicated Hardware for DL Are Still to Be Explored?: Current off-the-shelf mobile devices usually already have the strong computing capability at a low-end laptop. In addition, most of them have integrated various processors that are designed for different purposes, such as CPU, GPU, and DSP. While incorporating dedicated hardware for DL acceleration depends largely on hardware manufacture, it is more practical to have software accelerators on mobile devices to take full advantage of available hardware resources by using hardware and software codesign.

Another way to enable DL on mobile devices is to employ new hardware (e.g., mobile GPU [122], FPGA [123], [124], and application-specific integrated circuit (ASIC) [125]–[128]) dedicated to DL on mobile devices. So far, those designs require special manufacturing processes. Also, not many new hardware designs have been done for efficient DL-based applications.

To facilitate the analysis of DL optimization for mobile devices and applications, we propose a DL optimization pipeline, as shown in Fig. 5. In particular, we identify four major mobile applications that have the most extensive use of DL as the focus of this survey, namely image classification/object detection, speech/speaker recognition, activity recognition, and mobile security. While DL has already been adopted in a variety of applications to improve their performance, these applications cover the state-of-the-art DL design and optimization research. Overall, we categorize research on DL optimization for mobile devices and applications into two types: model-oriented optimization mechanisms and nonmodel-oriented optimization mechanisms. For most mobile applications, the first type can reduce DL models' size and inference time. These mechanisms include the optimum DL model selection, which explores DL models that are most suitable for a particular type of mobile application, and the DL model optimization, which develops compressing or pruning technologies to reduce the complexity of DL models for mobile deployment. In addition to the first type of optimization, the second type could further boost the performance of DL on mobile devices. Given a particular hardware environment, researchers have developed mobile DL software accelerators to decompose a DL model and deploy the decomposed components on the most suitable mobile processing hardware (e.g., CPU, GPU, and DSP). Except for software-oriented optimization approaches, there are research works focusing on mobile hardware designs for DL, which proposes special computer architectures enabling highly efficient DL training and inference with low energy consumption. Next, we introduce our findings in each component of the pipeline.

B. Model-Oriented Optimization Mechanisms for Mobile Devices and Applications

1) Optimum DL Model Selection: With the emerging use of mobile devices in AI or VR applications, research in the computer vision domain has led to the trend of using DL on mobile devices. DL models that are specially designed for mobile devices have shown the initial success in image classification and object detection whose performance is comparable to that of the state-of-the-art DL models (e.g., VGG [65] and AlexNet [66]) on personal computers.

a) DL Model Selection for Image Classification: Specifically, MobileNets [64] and SqueezeNet [129] have already shown their effectiveness on mobile devices with power constraints for image classification. In MobileNets, by using the width and resolution multipliers, researchers can trade off a reasonable amount of accuracy and model size to build smaller and faster versions. In particular,

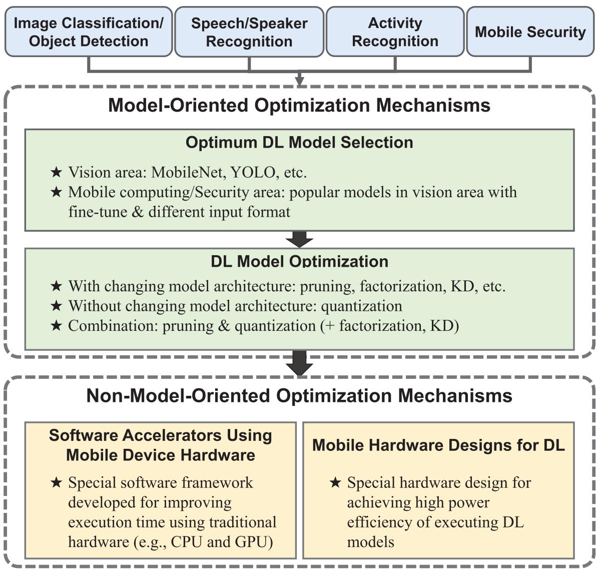


Image Classification/ Object Detection
Speech/Speaker Recognition
Activity Recognition
Mobile Security
Model-Oriented Optimization Mechanisms
Optimum DL Model Selection
★ Vision area: MobileNet, YOLO, etc.
★ Mobile computing/Security area: popular models in vision area with fine-tune & different input format
DL Model Optimization
★ With changing model architecture: pruning, factorization, KD, etc.
★ Without changing model architecture: quantization
★ Combination: pruning & quantization (+ factorization, KD)
Non-Model-Oriented Optimization Mechanisms
Software Accelerators Using Mobile Device Hardware
★ Special software framework developed for improving execution time using traditional hardware (e.g., CPU and GPU)
Mobile Hardware Designs for DL
★ Special hardware design for achieving high power efficiency of executing DL models

Fig. 5. ***DL optimization pipeline for mobile applications. We identify four major mobile applications that have the most extensive use of DL as the focus of this survey. Overall, we categorize research on DL optimization for mobile devices and applications into two types. The first one is model-oriented optimization mechanisms, which includes the optimum DL model selection and DL model optimization. The second one is nonmodel-oriented optimization mechanisms, which further boosts the performance of DL via software accelerators using mobile device hardware and mobile hardware designs for DL.***

MobileNet is nearly as accurate as VGG16 but 32 times smaller and 27 times less compute-intensive. Moreover, it is 4% better than AlexNet, yet 45 times smaller and 9.4 times less compute than AlexNet, while SqueezeNet achieves a model size less than 0.5 MB with 50 times fewer parameters than AlexNet, but with the same level of accuracy. It can be entirely deployed in the FPGA and eliminate the need for off-chip memory accesses to load model parameters.

b) DL Model Selection for Object Detection: As for object detection, a fast single-shot object detector, SSD [130], has been built on MobileNet, which can be deployed on the mobile and edge devices [131]. In particular, the prototype SSD could significantly outperform the state-of-the-art object detector counterparts (e.g., Faster R-CNN [132] and YOLOv1 [133]) in terms of both accuracy and speed. For example, the SSD512 (i.e., SSD with 512×512 input) model has 10% higher accuracy and three times faster than the Faster R-CNN. The SSD300 (i.e., SSD with 300×300 input) model could perform object detection at 59 frames/s, which is about three times faster than the current real-time YOLOv1 model and producing a superior detection accuracy of 74.3% on the Pascal VOC2007 dataset. Moreover, YOLOv2 [134] was developed as an enhanced version of YOLOv1 whose inference speed could be twice that of SSD. In particular, YOLOv2 added the batch normalization on all convolutional layers and adopted anchor boxes as SSD and Faster R-CNN do. It could also be run at a variety of image sizes to provide a smooth tradeoff between speed and accuracy, which has been widely tested on mobile devices for achieving real-time processing.

c) DL Model Selection for Activity Recognition and Security: While the DL models mentioned above are specially designed for computer vision-based applications, they could be adopted in other domains, such as activity recognition and security. There are two basic steps for adoption. First, according to different data types, it is necessary to change the input format for certain datasets (e.g., the activity data collected from accelerometer and gyroscope sensors in the UCI HAR [135] and WISDM datasets [136] and network traffic data collected from the UNSW-NB15 [137] and CIDDS-001 [138] datasets in security area). Second, base model's hyperparameters (e.g., number of neurons, convolution kernel width, and learning rate) should be tuned based on both accuracy and inference time requirement of different applications and the constraint of adopted mobile hardware. In particular, there are several benefits of using those state-of-the-art DL models in the vision field. First, those models have been publicly available for usage. Therefore, they only need to slightly change the input format. Second, adopting those models could provide a general evaluation framework among research work. Third, those models should be able to support real-time applications on mobile devices.

The next optimization option is to apply the general DL optimization mechanisms. In particular, DL model optimization aims at reducing the model size so as to fit into resource-constrained mobile devices. Such methods can be roughly divided into two categories: with changing model architecture and without changing model architecture, as shown in Table 3. The former reduces the model size by shrinking the original model architecture, while the latter shrinks the model weight size retaining the original model architecture.

2) DL Model Optimization With Changing Model Architecture: Several DL optimization methods in with changing model architecture have been proposed to change model architectures by either introducing new layers by factorization [139]–[141], pruning connections between neurons [1], [3], [39], [40], or even generating totally new architecture based on original models [144]–[146].

a) Factorization: Factorization [149] changes the underlying architecture by adding new layers. Specifically, it compresses the weight matrix W of an original DL model into two lower rank weight matrix W_a and W_b via singular value decomposition (SVD). As long as the summed parameters in two factorized matrices W_a and W_b are less than the original matrix W, the model size will be compressed. Furthermore, this technique can uniformly be applied to convolutional layers and fully connected layers. For example, DXTK [139] showed that the DNN model after factorization can run 9.7 times faster than its unmodified version. The latest work [150] showed that factorization could reduce the model size up to approximately 75%

Table 3 Classification of With-Changing-Model-Architecture and Without-Changing-Model-Architecture DL Model Optimization Methods

With-Changing-Model-Architecture	**Without-Changing-Model-Architecture**

Factorization [139]–[142];
Pruning [1], [3], [39], [40], [142], [143];
Generating new architecture [144]–[146];
Knowledge Distillation (KD) [144], [146];
	Quantization [40], [142], [143], [147], [148];

without decreasing the accuracy. However, factorization is computationally expensive and not suitable for large deep models since the computation cost increases exponentially as the layer increases.

b) Pruning: In addition to factorization, pruning could compress DL models by removing small-weight connections below a threshold. For example, Shi *et al.* [151] showed that the DNN model after pruning could achieve up to 25.6 times reduction on transmission workload, 6 times acceleration on total computation, and 4.81 times reduction on end-to-end latency compared to the original DNN model without pruning. DeepIoT [39] presented the performance of adopting pruning on DL structures for sensing applications, including fully connected network, CNN, and RNN, as well as their combinations. It reduced the size of deep neural networks by 90% to 98.9%, the execution time by 71.4% to 94.5%, and energy consumption by 72.2% to 95.7% without loss of accuracy. The results show the potential of pruning for deploying deep neural networks on resource-constrained embedded devices. Currently, Tensorflow and Pytorch support weight pruning for both convolution layers and fully connected layers. However, pruning increases training time and also generates a sparse matrix that requires some dedicated hardware.

c) Knowledge Distillation: Different from pruning and factorization, a knowledge distillation (KD) method [144] trains a small model (i.e., student model) based on large original models or whole ensemble of models (i.e., teacher model), This is also called a teacher–student network. Knowledge is the distribution of class probabilities predicted by the teacher model based on the softmax layer output. In particular, knowledge is transferred to the distilled model (i.e., compressed model) by training it to match each class probability in the output of softmax layer of teacher model. For example, Hinton *et al.* [144] showed that they could significantly improve the model of a heavily used commercial system by distilling the knowledge in an ensemble of models into a single model. In particular, more than 80% of the improvement in classification accuracy achieved by using an ensemble of ten DNN models in automatic speech recognition (ASR) was transferred to the distilled model. Ashok *et al.* [146] showed that KD could be combined with pruning to improve the compression rate of about two times than using pruning alone while also maintaining the accuracy of the teacher model. However, KD heavily relies on the softmax layer and has a strict assumption about the size and structure of the teacher–student network. As a result, this method is more suitable for small- and middle-size models instead of large models.

3) DL Model Optimization Without Changing Model Architecture: The DL optimization methods listed above can reduce the model size and retain the decent accuracy with changes in model architecture. Instead, the model optimization of without changing model architecture can reduce the model size without changing the original model architecture and have little degradation in model accuracy.

a) Quantization: It can use quantization to optimize the model by replacing the full-precision weight (i.e., 32 bit) into n-bit precision weight (e.g., binary quantization having 1-bit weight) and reducing the memory by a factor of $32/n$. Moreover, quantization can speed up inference since the costly multiplication and addition operations are replaced by cheap XNOR and bit-count operations of two quantized vectors with relatively lower energy requirements. For example, the system [147] showed that quantizing 16- or 32-bit weight to 4–10 bits could achieve energy reductions of up to 30 times without sacrificing algorithm performance. However, retraining is required after quantization and different layers may have different sensitivities to quantization resolution [148]. In addition, computer hardware (e.g., CPU and GPU) does not support all the quantization resolutions. For instance, Nvidia TensorRT only supports integer quantization and Float16 quantization. Currently, popular DL libraries (e.g., Tensorflow and Pytorch) support only integer quantization and Float16 quantization. A special kind of quantization model is binarized neural networks (BNNs) [152]–[154] that perform the pure-logical computation by utilizing 1-bit weights and activations. Due to the fewer memory access, it significantly speeds up inference time and reduces energy consumption in resource-constrained devices. For example, Ding *et al.* [154] used 1-bit weights and activations to compute the distribution loss for regularizing the activation flow and develop a framework to systematically formulate the loss. In addition, the proposed approach is robust to select the hyperparameters (i.e., learning rate and optimizer) to train the state-of-the-art networks (i.e., AlexNet) on the ImageNet dataset with high accuracy. The evaluation results demonstrate that the proposed BNN approach can realize efficient inference on resource-limited mobile devices with lower inference latency and energy consumption. Lin *et al.* [153] introduced two innovative optimization approaches to training BNNs at run time. The approach used a linear combination of multiple binary weight bases to approximate full-precision weights and employ multiple binary activations to migmatite information loss. The experimental result shows that the proposed approach significantly reduces the memory cost while maintaining high accuracy.

Thus, the BNN approach is a potential option to enable the network to deploy on resource-constrained mobile devices.

4) Combining DL Model Optimization With and Without Changing Model Architecture: Although the quantization technique can compress the network complexity by using limited numerical precision or clustering parameters, the compression ratio of quantization is usually less than the optimization methods with changing architecture such as pruning method. Since those two categories of methods are not conflicting with each other, some research [40], [142], [143] combined both of them for better performance. Deep compression [40] adopted both pruning and quantization to reduce the storage requirement of neural networks by 35–49 times without affecting their accuracy. It also achieved 3–4 times layerwise speedup and 3–7 times better energy efficiency. This allowed fitting the model into on-chip SRAM cache rather than off-chip DRAM memory, which facilitated the use of complex neural networks in mobile applications with limited memory size and download bandwidth. CLIP-Q [143] combined network pruning and weight quantization in a single learning framework to perform pruning and quantization jointly. It obtained the state-of-the-art compression rates of 51 times of AlexNet, 10 times of GoogLeNet, and 15 times of ResNet-50. Shi *et al.* [142] presented a compression approach based on the combination of low-rank matrix factorization and quantization training to reduce complexity for neural networks. For a three-layer LSTM network, the original model size could be reduced to 1% with negligible loss of accuracy. Moreover, factorization and KD can be further applied for reducing redundant computations and achieving faster inference.

In addition to the DL model optimization methods for reducing the model size, many existing works also focus on generic optimization methods. These optimization methods optimize the training algorithms, such as stochastic gradient descent (SGD), adaptive gradient methods, and distribution methods, to reduce the computational cost, energy consumption, and speed up the training process. They aim to solve the issues of gradient explosion, vanishing, and the undesirable spectrum during the training process meanwhile keeping high accuracy. For example, Le *et al.* [155] demonstrated that more sophisticated optimization methods (e.g., limited memory BFGS (L-BFGS) and conjugate gradient (CG) with line search) could significantly simplify and speed up the process of training algorithms compared to the relatively simple optimization methods (e.g., SGD). MENNDL [156] developed a genetic algorithm to automate network selection on computational clusters through hyperparameter optimization. It simplifies the process of applying deep networks to new applications and domains compared to the manual selection for the hyperparameter. To summarize, such generic optimization methods reducing the training iterations result in computational cost, memory accesses, and energy consumption reduction. Researchers could adopt these generic optimization methods combined with other optimization methods to enable the efficient training of the models on resource-constrained mobile devices.

C. Nonmodel-Oriented Optimization Mechanisms

In addition to the model-oriented optimization mechanisms, there are some nonmodel-oriented ones that could further boost the performance of DL on mobile devices.

1) Mobile DL Software Accelerators: It is a service to adopt pretrained DL models on mobile devices. It decomposes a deep model across a mix of heterogeneous processors (e.g., CPU, GPU, and DSP) to maximize energy efficiency and execution time, with limited mobile resources, such as computation power and memory size. It has several advantages. First, it removes the barriers preventing existing pretrained DL models from being adopted by current mobile and wearable devices. Moreover, it allows complex DL models to run on commodity mobile devices with acceptable resource consumption levels and low latency. There are several software accelerators to optimize the DL for mobile devices.

DeepX [19] took an important step toward adopting DL models to mobile and wearable devices. This software accelerator significantly lowered the device resources (i.e., memory, computation, and energy) required by DL that currently acts as a severe bottleneck to mobile adoption. Specifically, a pair of resource control algorithms were proposed for the inference stage of DL. First, DeepX decomposed monolithic deep model network architectures into various unit blocks, which were then executed by heterogeneous local device processors (e.g., GPUs and CPUs). Second, it performed principled resource scaling that adjusted the architecture of deep models to reduce the overhead that each unit block introduces. Experiments showed that across various model and processor combinations, the mean energy benefit of DeepX was 7.12 times (Snapdragon) and 26.7 times (Tegra) under the time requirement.

Another accelerator CNNdroid [157] was a GPU-accelerated library for execution of trained deep CNNs on Android-based mobile devices. It was compatible with CNN models trained by common libraries, such as Caffe, Torch, and Theano. Empirical evaluations showed that CNNdroid achieved up to 60 times faster speed and 130 times energy saving on current mobile devices. Moreover, the source code, documentation, and sample projects were publicly available (https://github.com/ENCP/CNNdroid).

Although DeepX and CNNdroid could use computing resources on mobile devices to accelerate DL models, there are some limitations. These frameworks require some dedicated hardware (e.g., GPU), which are not available for some mobile devices. In addition, they have not been integrated well with popular DL frameworks and thus need external tools to run models trained with existing libraries such as TensorFlow. Different from the above two methods,

Table 4 Summary of Software Accelerators

Accelerators	Design Principle	Advantages	Disadvantages
DeepX [19]	(1) Leverage a mix of heterogeneous processors (e.g., GPUs, LPUs); (2) Offer two resource control and optimization algorithms.	Decompose a deep model across available processors to maximize energy-efficiency and execution time.	DeepX is not applicable to those deep learning networks with temporal information.
CNNdroid [157]	Use parallel processing capabilities of mobile GPUs to accelerate deep CNN computations on current mobile devices.	(1) Compatible with CNN models trained by common desktop/server libraries; (2) Easy to configure and integrate into any Android app.	Do not consider the underlying hardware architecture to thoroughly make use of the hardware specifics.
RSTensorFlow [15]	Modify the kernels of TensorFlow operations to leverage the RenderScript heterogeneous computing framework on Android devices.	Utilize the power of available computation resources while running models trained with TensorFlow without requiring external tools.	Optimizing the energy costs of running on CPU vs GPU are potential future research directions.
DeepMon [18]	(1) Develop optimizations for processing convolutional layers; (2) Apply system-level optimizations to accelerate the matrix calculation.	Allow large DNNs to run on commodity mobile devices at a low latency.	Only extract features from video frames and need to be integrated with a complete cloud-enabled solution.
CADNN [36]	Leverage model compression to improve the DNN inference execution performance in the mobile environment.	Propose three major optimizations targeting modern mobile architectures.	Focus on the acceleration of existing DNN models designed for mobile applications.

RenderScript [15] was proposed as an extension to TensorFlow. By integrating the acceleration framework tightly into TensorFlow, researchers could easily take advantage of the heterogeneous computing resources (CPU and GPU) on mobile devices without the need for other external tools. DL models could run three times faster by using RenderScript.

Another approach, DeepMon [18], allowed large DNNs to run on commercial mobile devices at low latency. Prior work, such as DeepX, has focused on smaller DNNs and more powerful noncommodity mobile devices such as the Tegra K1. DeepMon could complement DeepX by supporting various mobile GPUs (e.g., Adreno 420, Adreno 430, and Mali T 880) on the market. Moreover, DeepMon proposed to reduce the latency of convolutional layers. In particular, by devising a suite of optimization techniques, including the convolutional layer caching, decomposition, and matrix multiplication optimizations, DeepMon is two times faster than DeepX.

Furthermore, Niu *et al.* [36] found that most of the existing accelerators, such as DeepMon and DeepX, did not explore possible optimization opportunities in computation and memory footprint reductions offered by model compression, including weight pruning and weight quantization. Therefore, a significant performance gap still remained between the peak performance that could be potentially offered by mobile devices and what the existing systems actually achieved. Based on that, Niu *et al.* [36] proposed CADNN, a programming framework, to efficiently execute DNN inference on mobile devices with the compression (i.e., pruning) and a set of architecture-aware optimizations (e.g., computation pattern transformation, redundant memory load elimination, smart selection of memory, and computation optimization parameters). Based on evaluation studies, CADNN is about 8.8 times and 6.4 times faster than TensorFlow Lite and TVM, respectively, two popular and highly optimized dense DNN execution frameworks.

As shown in Table 4, software accelerators can benefit from the deployments of pretrained DL models on mobile devices based on various design principles. If the researchers would like to deploy their DL models on some resource-constrained mobile devices (e.g., IoT) and improve the energy efficiency and lower the memory and computation requirements of the DL, some common accelerators with resource control strategies and optimization approaches (e.g., DeepX and CADNN) might be considered. If the pretrained DL model is developed with some specific libraries (e.g., TensorFlow) or is deployed on some specific devices (e.g., Android devices), some accelerators, such as RendorScript and CNNdroid, would benefit the researchers a lot since they aimed at specific platforms or frameworks. If the researchers would like to adopt some large-scale DNN models, DeepMon is a better choice since it helps adopt large DNN models with low latency on various mobile devices in the market.

2) Mobile Hardware Designs for DL: Advances in DL are mainly due to powerful computer systems. Compared to such systems, mobile devices have limited computational power and battery lives, making them not suitable for DL deployment. In existing research, the most common computing processors used for DL on mobile devices are CPUs, GPUs, FPGA, and ASIC. We compare the power efficiency, time/cost budget, and compatibility of these processors in Fig. 6. CPU and mobile GPU are widely used in the embedded platform and mobile devices due to their high availability and compatibility. They can be supported by multiple DL frameworks and various programming languages. Furthermore, some GPU-based execution frameworks [158], [159] can be utilized to provide low latency, high throughput, and efficient on-chip resource utilization. It is a good choice for the researchers to start with CPUs and GPUs to explore the feasibility of deploying DL models on mobile devices. However, as shown in Fig. 6, CPUs and GPUs have the lowest power efficiency, which is not really mobile-friendly. To enable DL on power-constrained mobile devices, ASIC is the most power-efficient architecture suitable for processing DL on mobile devices. Compared to ASIC, FPGA is a balanced choice considering the power efficiency, compatibility, and developing budget. FPGA is also more flexible in development since it can be (re)programmed by firmware and supported by open-sourced libraries (e.g., Caffe).

Recent studies have shown that highly efficient DL training and inference with low energy consumption could be achieved using processors with specially designed architecture. ASIC [125]–[128], [160], [161] and FPGA [123], [124], [162]) have offered a good starting point for developing DL processors for mobile devices. We summarized these works in Table 5. For instance,

Table 5 Summary of Hardware Designs

Accelerators	Categories	Advantages	Disadvantages
TrueNorth [125]	ASIC-based	Have associated software to aid in development.	The maximum matrix that one synaptic core can handle is 256 × 256, and it supports only a simplified leaky–integrate-and-fire neuron model.
VPU [126]	ASIC-based	It can be deployed onboard a drone and used to perform inference on drone imagery, without significantly taxing system resources.	The execution time per inference using one chip is 4× slower compared to a reference CPU/GPU implementation.
EIE [127]	ASIC-based	Operating directly on compressed networks enables the large neural network models to fit in on-chip SRAM, which results in 120× better energy savings compared to accessing from external DRAM.	Not scalable due to all-to-all processing element broadcasts and a BW link of one element per cycle.
DianNao [128]	ASIC-based	The accelerator achieves high throughput in a small area, power and energy footprint.	The scalability and efficiency are severely limited by the bandwidth constraints of the memory system.
Zhang et al. [123]	FPGA-based	Successfully deploy the deeper VGGNet into an embedded FPGA platform, with several optimization techniques.	1) This approach is limited to VGG models. 2) Not flexible enough to match each layer's distinct features and result in underutilization of hardware resources.
FPGA15 [124]	FPGA-based	Under the constraints of computation resource and memory bandwidth, it explores all possible solutions in the design space using a roofline model.	The accelerator design was only applied to several CONV layers rather than the full CNN.

TrueNorth [125] proposed a real-time neurosynaptic processor for mobile devices based on ASIC, which achieved an extremely low typical power consumption of 65 mW. Besides, TrueNorth was fully configurable in terms of connectivity and neural parameters to allow custom configurations for a wide range of applications. The authors also introduced an asynchronous–synchronous tool that may facilitate general ASIC design. An ASIC-based VPU [126] has been used to enable low-power DL computation. This article showed that VPU, as a highly parallel vector coprocessor with low power consumption, could reduce the thermal design power (TDP) up to eight times compared to their CPU and GPU implementations. Han *et al.* [127] proposed an optimized energy-efficient engine (EIE) to operate on compressed deep neural networks. By leveraging sparsity in both the activations and weights and integrating weight sharing and quantization techniques, this engine outperformed CPU, GPU, and mobile GPU by the factors of 189×, 13×, and 307× and consumed 24 000×, 3400×, and 2700× less energy than CPU, GPU, and mobile GPU, respectively. Chen *et al.* [128] proposed an ASIC accelerator, DianNao, for the fast and low-energy execution of the inference of large CNNs and DNNs in a small form factor. This work had a special emphasis on the impact of memory on accelerator design, performance, and energy. This accelerator had high throughput, capable of performing 452 GOP/s in a small footprint of 3.02 mm and 485 mW. It achieved a speedup of 117.87× and an energy reduction of 21.08× over a 128-bit 2-GHz single instruction multiple data (SIMD) core with a normal cache hierarchy.

Qiu *et al.* [123] went with an FPGA-based approach for accelerating the large-scale image classification using a CNN. A state-of-the-art CNN, VGG16-SVD, was implemented on an FPGA platform. Experimental results showed that GPU consumed 26 times more power consumption than the proposed FPGA hardware. Zhang *et al.* [124] proposed a roofline-model-based FPGA accelerator. In this method, they first optimized CNN's computation and memory access. Then, they modeled all possible designs in the roofline model to find the best design for each layer. They realized an implementation on the Xilinx VC707 board and achieved a 4.8× speedup over CPU.

Based on the proposed DL optimization pipeline for mobile applications, when deploying DL models on common mobile and embedded devices, researchers could apply different optimization mechanisms to improve the performance of mobile applications. The model-oriented optimization approaches, such as pruning, could be leveraged to reduce the memory and storage requirements and speed up the inference time. The nonmodel-oriented optimization mechanisms, such as software accelerators, could further improve the efficiency of executing DL operations without modifying models. While the improvement of software accelerators may be limited by various software environments such as operating systems and drivers, hardware accelerators based on special computer architecture (e.g., ASIC or FPGA) could be designed to improve the efficiency of DL operations on mobile devices significantly when time and money budget is sufficient.

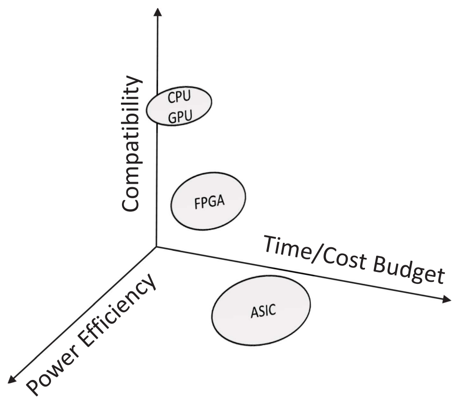


Fig. 6. *Metrics of different computation components. There are three metrics we distinguish on various hardware devices, including power efficiency, compatibility, and time/cost budget.*

IV. SOFTWARE LIBRARIES

This section summarizes software libraries regarding their support for software optimization and hardware optimization, as shown in Fig. 7. For software optimiza-

Table 6 Software Optimization Support

	General DL Optimization Methods				Software Accelerators
	pruning	**quantization**	**factorization**	**KD**	
TensorFlow	weight pruning*	Integer quantization*, float16 quantization*	✓	✓	DeepX, RenderScript, DeepMon
Caffe2	N/A	Integer quantization*	✓	✓	DeepX, CNNdroid
Keras	weight pruning*	Integer quantization*, float16 quantization*	✓	✓	DeepX
Pytorch	weight pruning	Integer quantization*	✓	✓	DeepX, CNNdroid
MXNET	N/A	Integer quantization*	✓	N/A	N/A
DL4j	N/A	N/A	N/A	N/A	DeepX

tion, we discuss popular model architectures and general DL optimization approaches (e.g., quantization and pruning) as well as software accelerators. For hardware optimization, we summarize the popular libraries for various types of hardware such as mobile and powerful embedded devices and microcontroller (i.e., the IoT devices). Besides, we also consider the related designs such as ASIC and FPGA.

A. Software Optimization Support

Popular DL libraries, such as Tensorflow, Pytorch, Keras, Caffe2, and Deeplearning4j (DL4j), support most of the state-of-the-art model architectures in various application domains, as shown in Tables 1 and 2. In particular, some pretrained models for image, text, and video applications, such as Mobilenet, Resnet, Inception, Convnet, and VGG, are available [163]–[165] and fine-tuning in user-specific applications for downloading.

Besides supporting model architectures, those libraries also provide general optimization procedures. As shown in Table 6, Tensorflow, Pytorch, and Keras currently support common optimization algorithms via both quantization and pruning [166]. In particular, Tensorflow officially provides full integer quantization, float16 quantization, and magnitude-based weight pruning. For example, it can reduce the model size fourfold and increase the computational speed threefold on CPU, edge tensor processing unit (TPU), and microcontrollers via full integer quantization [167]. Moreover, its supported float16 quantization can reduce the model size twofold on CPU with potential GPU acceleration [167]. In addition, Tensorflow achieves model sparsity by supporting magnitude-based weight pruning [168]. Since Keras can adopt Tensorflow as backend, it can carry those optimization functionalities provided by Tensorflow. In contrast, PyTorch supports only integer quantization allowing a four-time reduction in the model size and memory bandwidth requirements [169]. Also, it provides pruning capability via the third part package (e.g., Intel AI Lab proposes Distiller [170], an open-source Python package for neural network compression research). Caffe2 and MXNet currently support only integer quantization via the official API [171] and Apache/MXNet toolkit, respectively. Optimization techniques, such as factorization and KD, can also be adopted when using those libraries. In addition, some existing software accelerators (e.g., DeepX, RenderScript, DeepMon, and CNNdroid) can also be adopted to further enhance computational performance by leveraging various mobile hardware components, such as GPU and LPU. Table 6 summarizes the accelerators and related libraries.

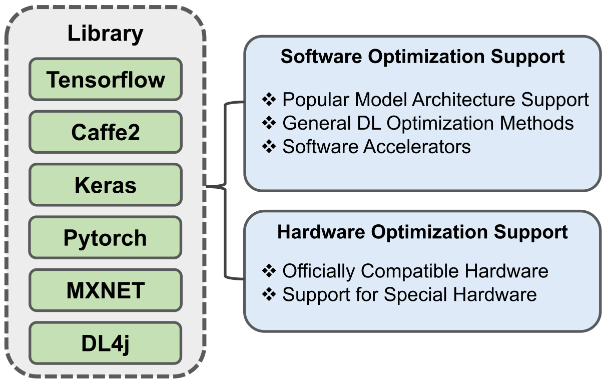

Library
Tensorflow
Caffe2
Keras
Pytorch
MXNET
DL4j
Software Optimization Support
❖ Popular Model Architecture Support
❖ General DL Optimization Methods
❖ Software Accelerators
Hardware Optimization Support
❖ Officially Compatible Hardware
❖ Support for Special Hardware

Fig. 7. ***Library optimization support. We summarize software libraries regarding their support for software optimization and hardware optimization. In software optimization, we discuss popular model architectures and general DL optimization methods as well as software accelerators. In hardware optimization, we summarize the popular libraries for various types of hardware. Besides, we also consider special designs, such as ASIC and FPGA.***

B. Hardware Optimization Support

Different hardware can support various DL frameworks and libraries, which made it convenient for researchers to deploy DL on their devices. Mobile phones and powerful embedded devices are the most common authorized and supported mobile hardware to deploy DL. Specifically, Android and iOS mobile phones are supported by most of the popular libraries, as shown in Table 7. In addition, TensorFlow, Caffe2, PyTorch, MXNet, and so on are available in the embedded device, such as Raspberry Pi, NVIDIA Tegra, and Nvidia Jetson TX2. Some libraries have been adopted for resource-constrained mobile devices such as microcontrollers (IoT). For example, TensorFlow Lite, a lightweight version of TensorFlow (https://www.tensoflow.org/lite/microcontrollers), supports the processors based on the Arm Cortex-M Series and ESP32 architecture (e.g., Arduino Nano 33 BLE Sense

Table 7 Hardware Optimization Support

	Official Supported Hardware		Special Hardware Design
	Mobile & powerful embedded device	**Microcontroller support (IoT)**	
TensorFlow	standard TensorFlow Lite (smartphone and Raspberry Pi)	TensorFlow Lite for Microcontrollers (processors based on the Arm Cortex-M Series and ESP32 architecture, e.g., Arduino Nano 33 BLE Sense, SparkFun Edge)	ASIC support
Caffe2	iOS, Android,Tegra, and Raspberry Pi platforms	CMSIS-NN optimized libraries for Arm Cortex-M processor cores	N/A
Keras	mobile and embedded devices	N/A	N/A
PyTorch	iOS, Android, and Raspberry Pi	N/A	ASIC support
MXNet	iOS, Android, Raspberry Pi, and NVIDIA Jetson TX	N/A	ASIC support
DL4j	Android, iOS	N/A	N/A

and SparkFun Edge). Caffe2 facilitates certain microcontroller (e.g., Arm Cortex-M processor cores) by using the CMSIS-NN optimized libraries. Furthermore, researchers also design customized hardware (e.g., ASIC and FPGA) to implement libraries, such as TensorFlow, Python, and MXNet.

C. Model Conversion Tools

There are many available libraries and each of them has its advantages and use cases. To make the best use of different features provided by these libraries and allow researchers to take full advantage of the libraries based on their situation, model conversion tools are developed. Then, users can smoothly convert DL models from one library to another library without reprogramming. For example, MMdnn [172] is a comprehensive and cross-framework tool to convert, visualize, and diagnose DL models. It is a universal converter to convert DL models among a number of existing frameworks, making a trained model of one framework be easily deployed on another framework for inference and prediction. Open neural network exchange (ONNX) [173] is an open standard format for representing machine learning models. It is facilitated by a community of partners for implementations in many frameworks, such as PyTorch, MXNet, Caffe2, and TensorFlow.

V. FL ON MOBILE DEVICES

Training DL network on mobile device is another challenging issue. FL is a way of enabling DL training by leveraging many client devices (e.g., mobile devices and the IoT devices) in a network. This solves the issue that a single resource-constrained device may never support training a DL model due to its limited hardware resources. In particular, all mobile devices need to devote their storage and computing resources for their data training. However, this requirement is not always satisfied. When FL deploys identical neural network models to heterogeneous devices, the ones with weak computational capacities may significantly delay the synchronous parameter aggregation [174]. Besides, a large amount of memory and power are needed for the training of advanced DL models and the storage of model parameters [175]. Thus, many aspects still need to be carefully considered, including memory limitation, energy budget, communication, synchronization, resource distribution, and privacy. In this work, we focus on the on-device optimization in the FL domain and mainly discuss how to save memory usage and energy consumption using the proposed DL optimization pipeline. Other aspects are out of the scope of discussion of this article.

A. Model-Oriented Optimization Mechanisms in FL

1) DL Model Selection for FL: Since the FL runs on multiple heterogeneous devices, which vary in computing ability and memory size, it is critical to first select the suitable models that are smoothly executed on various devices. Specifically, due to the limited memory and computing resources of some resource-constrained FL clients, the large size of CNN models (i.e., VGG and ResNet) is hardly adopted on them. Researchers have been exploring lightweight and sparse DL architecture for more general applicability. For instance, a novel trilayer FL scheme [176] is proposed to consume less amount of resources to accomplish a target convergence. A variant of the LSTM recurrent neural network called the coupled input and forget gate (CIFG) [177] is adopted for mobile keyboard predication with FL. The CIFG architecture is advantageous for the mobile device environment because the number of computations and the parameter set size are reduced with no impact on model performance. Moreover, a personalized FL framework (e.g., three-layer CNN) [178] is advocated to cope with the heterogeneity issues in FL environments.

2) DL Model Optimization for FL: Since some FL clients may not have enough memory size and energy budget to meet the system requirements for training, the researchers have been exploring how to reduce the memory cost and save the energy cost in FL training and prolong battery-power life duration. To address these issues, some researchers have adopted the with-changing-model-architecture model optimization approaches. For example, PruneFL [179] proposes an adaptive and distributed parameter pruning approach, which adapts the model size during FL to reduce both communication and computation overhead and minimize the overall training time while maintaining a similar accuracy as the original model. To reduce the memory requirement for training, Helios [174] devises a soft-training method to dynamically compress the original training model into the expected size through a rotating neuron training approach. FedPARL [176] reduces the model size by performing sample-based pruning, which is demonstrated useful for resource-constrained IoT devices. Moreover, without-changing-model-architecture model optimization approaches have also been adopted. For instance, Wu *et al.* [180] proposed a quantization framework that constrains all layers to low-bit width integers in both training and inference. This approach can reduce the energy of about five times compared to 32-bit floating-point operations (FLOPs). In addition, dynamic computation approaches are explored to optimize the model during the training and inference phases. The basic idea of those approaches is to only activate the partial neural network to process the input. For example, dynamic channel pruning techniques [181], [182] are adopted to identify channels in the deep neural network that are considered as unimportant and skip their computations. Some researchers [182], [183] apply dynamic layer skipping techniques on ResNet models and RNN models with skip connections.

B. Nonmodel-Oriented Optimization Mechanisms in FL

1) Mobile DL Software Accelerators for FL: Several possible solutions should be considered to facilitate on-device DL learning for FL systems. One direction is to improve DL hardware usage on mobile devices via the mobile DL software accelerators [15], [18], [19], [36], [157] as we discussed in Section III. Since these DL software accelerators can decompose a deep model across a mix of heterogeneous processors (e.g., CPU, GPU, and DSP) to maximize energy efficiency and execution time, with limited mobile resources (e.g., computation power and memory size), it removes the barriers preventing existing pretrained DL models from being adopted by current mobile devices in FL systems. For example, DeepX [19] significantly lowers the device resources (i.e., memory, computation, and energy) required by DL that currently acts as a severe bottleneck to mobile adoption through a pair of resource control algorithms, designed for the inference stage of DL. Experiments demonstrate its superior performance in terms of low execution time and energy consumption in DL running compared to cloud offloading-based approaches. This research is promising and likely to facilitate the development of sensor processing and mobile DL inference, which would enable on-device FL implementation at scale.

2) Mobile Hardware Designs for FL: Another direction is mobile hardware designs for FL. In recent years, few hardware designs are emerging to solve some specific problems in the FL area. For example, the complicated operations and large operands of privacy-preserving mechanisms (e.g., homomorphic encryption) impose significant overhead on FL. Maintaining accuracy and security more efficiently has been a key problem of FL. Yang *et al.* [184] proposed a hardware solution to accelerate the training phase in FL by designing an FPGA-based homomorphic encryption framework. Experiments show that the proposed accelerator achieves a near-optimal execution clock cycle, with a better digital signal processing efficiency, and reduces the encryption time by up to 71% than the existing designs. Another work [122] proposes HAFLO, a GPU-based solution to improve the performance of FL by reducing the significant computational overhead of homomorphic encryption. The core idea of HAFLO is to summarize a set of performance-critical homomorphic operators used by federated logistic regression and accelerate the execution of these operators through joint optimization of storage, IO, and computation. Experimental results show that the proposed method achieves a 49.9× speedup acceleration on a popular FL framework. Furthermore, as we discussed in Section III, some studies have demonstrated that highly efficient DL training and inference with low energy consumption could be achieved using specially designed processors. Some ASICs [125]–[128], [160], [161] and FPGAs [123], [124], [162] have offered a good starting point for developing DL processors to mitigate the constraints of hardware, memory, and power resources of mobile devices. These works are promising and likely to further enable on-device FL implementation at scale. Exploring how these works execute in FL systems might help tailor these solutions to solve the existing issues in FL.

C. Cooperative DNN Inference in FL

A fast enough DNN inference with acceptable resource consumption is also an essential consideration in FL due to the latency requirements of many applications and the resource constraints of a single device. The existing researchers have been taking advantage of the aggregated computational power of a cluster of mobile devices, coordinating them for faster DNN inference. For example, DeepThings [110] proposes an FTP method for dividing convolutional layers into independently distributable tasks and parallelizing them on multiple devices. It provides scalable CNN inference speedups of 1.7×–3.5× on 2–6 edge devices. MoDNN [185] partitions trained DNN models on several mobile devices, assigning more workload to the more powerful devices for DNN inference acceleration. It can accelerate the DNN computation by 2.17×–4.28× when the number of devices increases from 2 to 4. Hadidi *et al.* [186] adopted both data parallelism and model parallelism to accelerate DNN inference using several robots. CoEdge [187] dynamically partitions the DNN inference workload based on the computing capabilities of devices and network conditions. It achieves up to 4.49×–7.21× latency speedup over the local approach and 25.5%–66.9% energy reduction for four widely adopted CNN models. These works demonstrate that cooperative DNN inference in FL could significantly speed up model inference and reduce resource consumption on every mobile device.

VI. POTENTIAL RESEARCH OPPORTUNITIES IN MOBILE APPLICATIONS

In this section, we provide some potential research opportunities for mobile applications in the areas of database and model, inference time, power consumption, hardware design, and optimization tradeoff on different hardware.

A. Database and Model

As discussed in Section II, the existing DL work in nonvision fields usually adopts various lightweight DL models. There is also a lack of baseline models in the nonvision field based on which to compare with each other's performance. Therefore, it is desirable to select general DL models as the basis for performance evaluation. In addition, given the fact that various types of mobile devices could be used, a public repository of benchmark performance in various types of hardware could be of great benefit to the research community. Moreover, the nonvision fields still lack comprehensive, diverse, and high-quality datasets for DL model training and evaluation. There are two reasons for such inadequacy. First, service providers prefer to keep their data confidential and rarely publish their datasets. Second, due to limitations of hardware resource and network conditions, data collected from mobile devices usually come with loss, redundancy, mislabeling, and class imbalance. To promote adopting DL on mobile devices for various application domains in both academic and industry communities, researchers and companies are encouraged to collect and publish more high-quality datasets.

B. Inference Time

As discussed in Section II, the existing work using DL in nonvision fields (e.g., activity recognition and mobile security) focuses much more on accuracy than inference time. However, those applications require certain speed for inference in reality. For example, human daily activities recognition requires the inferencing time (about 0.3–3 s) to achieve real-time nature. In the security area, a detecting time with a delay of 2 s is usually considered as acceptable for detecting attacks [188], [189]. Therefore, researchers should also evaluate inference time for those nonvision applications in the future. Besides, since different mobile hardware have different computational power, researchers may consider an alternative way (e.g., FLOPs widely used in the vision field) to fairly evaluate the potential inference speed of their systems. In addition, improving the inference speed is also critical for mobile devices. In particular, developing and adopting various compression techniques to reduce the inference time on mobile devices still need exploration.

C. Power Consumption

With the aid of highly effective computation components (e.g., GPU and ASIC) in mobile devices, DL models can be easily deployed with decent performances [190]. However, since DL is computationally intensive, energy and capability constraints should be considered when we deploy DL models on mobile devices because of their limited battery capacities. As a result, reducing the energy consumption of mobile devices with respect to data collection (e.g., from motion sensors and GPS) in motion sensor-based applications might be an open research area. Another interesting research direction is to optimize energy consumption when deploying DL models on mobile devices. Although designing some specific hardware chips (e.g., FPGAs and ASICs) can achieve this purpose in Section III, the generality of these special hardware design, as well as their interoperability and compatibility with existing hardware platforms, remains as challenges.

D. Hardware Design

DL on mobile devices is facilitated by mobile GPUs but is usually limited within several layers, as discussed in Section II, due to the constraint of computation and memory resources. Thus, the performance of some mobile applications is usually not satisfactory. In future research, in order to execute more complex models with better performance, mobile GPU with more CUDA cores and large graphic memories needs to be developed. In addition to powerful hardware components, some algorithms, such as the Toeplitz matrix, and Winograd and Strassen algorithms, can be explored to improve the computing performance by taking advantage of the low-latency temporary storage architecture of GPU [191].

Compared with powerful mobile devices, such as smartphones, the resource bottleneck in microcontrollers is more serious. Specifically, smartphones can now equip with several gigabytes of RAM, but microcontrollers, such as the ARM Cortex series, are limited to just hundreds of kilobytes. Some techniques, such as binary deep architectures, have the potential to fill this gap [192]. Such architectures could not only build extremely small models but also remove the requirements for expensive multiplication operations. Besides, the limited on-chip memory capacity often causes massive off-chip memory access and leads to very high energy consumption. Some frameworks, such as retention-aware neural acceleration (RANA) [193], can be explored to save the energy consumption of microcontrollers.

Furthermore, FPGAs and ASICs are also promising for DL. However, they currently lack adequate software support to fully achieve their potential in DL. Since FPGAs and ASICs are built with spatial architectures with low-energy on-chip memory, in future research, reusable dataflow algorithms can provide solutions to reduce data movements. Moreover, a promising field called neuromorphic computing enables information processing at meager energy cost on electronic devices via emulating the electrical behaviors of biological neural networks in human brains. Based on this technique, some companies are trying to develop ASIC chips, such as TrueNorth from IBM. Since neuromorphic architectures are more suitable for brain-like computations and achieve decent power efficiency, as discussed in Section III, they would be an attractive topic for future research.

E. Optimization Tradeoff on Different Hardware

In DL, larger and deeper models usually produce higher accuracy. However, they may also lead to larger inference time and more energy consumption. Building DL models on mobile devices with an accurate scheme, fast inference time, and without mass power consumption and huge memory usage should be a good research direction. Besides, even with various existing optimization approaches for DL on mobile platforms, it is still challenging to adapt DL models to various types of device hardware, while conventional optimization methods always ignore different hardware architectures and optimize all the DL models in a uniform way. Recent work [194] has highlighted that pruning and quantization methodologies relied on formulations are hardware-unaware, and they do not necessarily result in optimal configurations in terms of hardware efficiency. In order to solve the above issues, some new techniques, such as design automation technique [126], hardware-aware modeling methodologies [125], and stochastic computing [160], could be explored to achieve optional optimization on different hardware. The insights of these design policies will inspire future research in optimization with various types of mobile hardware to achieve efficient DL computing.

VII. CONCLUSION

This article provides a comprehensive review of the recent advancements of DL on mobile devices. Applications in various areas are summarized and demonstrated at the intersection of DL and mobile computing. The challenges of bringing DL on mobile devices are discussed, and a thorough DL optimization pipeline is introduced. Compared with the existing work, optimization approaches, including general and optional optimization approaches, are studied in detail. Moreover, the popular DL libraries are summarized with respect to software and hardware optimization support. These resources may serve as references and recommendations for researchers when they try to find appropriate optimization approaches and suitable DL libraries to deploy DL on mobile devices. Furthermore, the potential research opportunities of DL on mobile devices are also discussed with respect to database and model, inference time, power consumption, as well as hardware design, and optimization for tradeoff on different hardware. With the fast advances of DL algorithms and sustained emergence of powerful mobile hardware, DL on mobile devices would remain as a hot topic, and more challenging and interesting research directions would spring up in the future. ■

REFERENCES

[1] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, “AMC: AutoML for model compression and acceleration on mobile devices,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 784–800.

[2] K. Yang *et al.*, “CDeepArch: A compact deep neural network architecture for mobile sensing,” *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 2043–2055, Oct. 2019.

[3] D. Li, X. Wang, and D. Kong, “DeepRebirth: Accelerating deep neural network execution on mobile devices,” in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2322–2330.

[4] K. Yanai, R. Tanno, and K. Okamoto, “Efficient mobile implementation of a CNN-based object recognition system,” in *Proc. 24th ACM Int. Conf. Multimedia*, Oct. 2016, pp. 362–366.

[5] M. Tan *et al.*, “MnasNet: Platform-aware neural architecture search for mobile,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2820–2828.

[6] R. J. Wang, X. Li, and C. X. Ling, “Pelee: A real-time object detection system on mobile devices,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1963–1972.

[7] Y. He, Z. Pan, L. Li, Y. Shan, D. Cao, and L. Chen, “Real-time vehicle detection from short-range aerial image with compressed MobileNet,” in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 8339–8345.

[8] H. Ai, W. Xia, and Q. Zhang, “Speaker recognition based on lightweight neural network for smart home solutions,” in *Proc. Int. Symp. Cyberspace Saf. Secur.*, Cham, Switzerland: Springer, 2019, pp. 421–431.

[9] X. Lei, A. Senior, A. Gruenstein, and J. Sorensen, “Accurate and compact large vocabulary speech recognition on mobile devices,” in *Proc. Interspeech*, Aug. 2013, pp. 1–4.

[10] I. McGraw *et al.*, “Personalized speech recognition on mobile devices,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 5955–5959.

[11] Y. He *et al.*, “Streaming end-to-end speech recognition for mobile devices,” in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 6381–6385.

[12] F. Cruciani, I. Cleland, C. Nugent, P. McCullagh, K. Synnes, and J. Hallberg, “Automatic annotation for human activity recognition in free living using a smartphone,” *Sensors*, vol. 18, no. 7, p. 2203, Jul. 2018.

[13] T. Zebin, P. J. Scully, N. Peek, A. J. Casson, and K. B. Ozanyan, “Design and implementation of a convolutional neural network on an edge computing smartphone for human activity recognition,” *IEEE Access*, vol. 7, pp. 133509–133520, 2019.

[14] N. Mairittha, T. Mairittha, and S. Inoue, “On-device deep learning inference for efficient activity data collection,” *Sensors*, vol. 19, no. 15, p. 3434, Aug. 2019.

[15] M. Alzantot, Y. Wang, Z. Ren, and M. B. Srivastava, “RSTensorFlow: GPU enabled TensorFlow for deep learning on commodity Android devices,” in *Proc. 1st Int. Workshop Deep Learn. Mobile Syst. Appl. (EMDL)*, 2017, pp. 7–12.

[16] S. Tuli, N. Basumatary, and R. Buyya, “EdgeLens: Deep learning based object detection in integrated IoT, fog and cloud computing environments,” 2019, *arXiv:1906.11056*.

[17] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J. Oh, “Semisupervised deep reinforcement learning in support of IoT and smart city services,” *IEEE Internet Things J.*, vol. 5, no. 2, pp. 624–635, Apr. 2018.

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

[19] N. D. Lane *et al.*, “DeepX: A software accelerator for low-power deep learning inference on mobile devices,” in *Proc. 15th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2016, pp. 1–12.

[20] A. Antoniou and P. Angelov, “A general purpose intelligent surveillance system for mobile devices using deep learning,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 2879–2886.

[21] A. Xiao, R. Chen, D. Li, Y. Chen, and D. Wu, “An indoor positioning system based on static objects in large indoor scenes by using smartphone cameras,” *Sensors*, vol. 18, no. 7, p. 2229, Jul. 2018.

[22] N. Parikh, I. Shah, and S. Vahora, “Android smartphone based visual object recognition for visually impaired using deep learning,” in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2018, pp. 420–425.

[23] V. Mandal, L. Uong, and Y. Adu-Gyamfi, “Automated road crack detection using deep convolutional neural networks,” in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 5212–5215.

[24] L. Tobias, A. Ducournau, F. Rousseau, G. Mercier, and R. Fablet, “Convolutional neural networks for

object recognition on mobile devices: A case study," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 3530–3535.

[25] S. Bazrafkan, T. Nedelcu, P. Filipczuk, and P. Corcoran, "Deep learning for facial expression recognition: A step closer to a smartphone that knows your moods," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2017, pp. 217–220.

[26] I. Song, H.-J. Kim, and P. B. Jeon, "Deep learning for real-time robust facial expression recognition on a smartphone," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2014, pp. 564–567.

[27] K. Z. Haider, K. R, Malik, S. Khalid, T. Nawaz, and S. Jabbar, "Deepgender: Real-time gender classification using deep learning for smartphones," *J. Real Time Image Process.*, vol. 16, no. 1, pp. 15–29, Feb. 2019.

[28] B. A. Ashqar and S. S. Abu-Naser, "Image-based tomato leaves diseases detection using deep learning," *Int. J. Acad. Eng. Res.*, vol. 2, no. 12, pp. 10–16, Dec. 2018.

[29] Y. Gu, Q. Wang, and S. Kamijo, "Intelligent driving data recorder in smartphone using deep neural network-based speedometer and scene understanding," *IEEE Sensors J.*, vol. 19, no. 1, pp. 287–296, Jan. 2019.

[30] H. Maeda, Y. Sekimoto, and T. Seto, "Lightweight road manager: Smartphone-based automatic determination of road damage status by deep neural network," in *Proc. 5th ACM SIGSPATIAL Int. Workshop Mobile Geographic Inf. Syst.*, Oct. 2016, pp. 37–45.

[31] X. J. Tang, Y. H. Tay, N. A. Siam, and S. C. Lim, "MyWood-ID: Automated macroscopic wood identification system using smartphone and macro-lens," in *Proc. Int. Conf. Comput. Intell. Intell. Syst. (CIIS)*, 2018, pp. 37–43.

[32] A. Rattani and R. Derakhshani, "On fine-tuning convolutional neural networks for smartphone based ocular recognition," in *Proc. IEEE Int. Joint Conf. Biometrics (IJCB)*, Oct. 2017, pp. 762–767.

[33] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, "Road damage detection and classification using deep neural networks with smartphone images," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 33, no. 12, pp. 1127–1141, Jun. 2018.

[34] N. Hnoohom and S. Yuenyong, "Thai fast food image classification using deep learning," in *Proc. Int. ECTI Northern Sect. Conf. Electr., Electron., Comput. Telecommun. Eng. (ECTI-NCON)*, Feb. 2018, pp. 116–119.

[35] G. G. D. Angelo, A. G. C. Pacheco, and R. A. Krohling, "Skin lesion segmentation using deep learning for images acquired from smartphones," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[36] W. Niu, X. Ma, Y. Wang, and B. Ren, "26ms inference time for ResNet-50: Towards real-time execution of all DNNs on smartphone," 2019, *arXiv:1905.00571*.

[37] A. Alfarrarjeh, D. Trivedi, S. H. Kim, and C. Shahabi, "A deep learning approach for road damage detection from smartphone images," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 5201–5204.

[38] G. Mittal, K. B. Yagnik, M. Garg, and N. C. Krishnan, "SpotGarbage: Smartphone app to detect garbage using deep learning," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Sep. 2016, pp. 940–945.

[39] S. Yao, Y. Zhao, A. Zhang, L. Su, and T. Abdelzaher, "DeepIoT: Compressing deep neural network structures for sensing systems with a compressor-critic framework," in *Proc. 15th ACM Conf. Embedded Netw. Sensor Syst.*, Nov. 2017, pp. 1–14.

[40] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[41] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," 2016, *arXiv:1606.02147*.

[42] C. N. Duong, K. G. Quach, I. Jalata, N. Le, and K. Luu, "MobiFace: A lightweight deep learning face recognition on mobile devices," 2018, *arXiv:1811.11080*.

[43] R. Nagpal *et al.*, "Real-time traffic sign recognition using deep network for embedded platforms," *Electron. Imag.*, vol. 2019, no. 15, pp. 1–33, 2019.

[44] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

[45] Z. Qin *et al.*, "ThunderNet: Towards real-time generic object detection on mobile devices," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6718–6727.

[46] A. Sehgal and N. Kehtarnavaz, "A convolutional neural network smartphone app for real-time voice activity detection," *IEEE Access*, vol. 6, pp. 9017–9026, 2018.

[47] G. S. Bhat, N. Shankar, C. K. A. Reddy, and I. M. S. Panahi, "A real-time convolutional neural network based speech enhancement for hearing impaired listeners using smartphone," *IEEE Access*, vol. 7, pp. 78421–78433, 2019.

[48] J. Y. Han, W. Z. Zheng, R. J. Huang, Y. Tsao, and Y. H. Lai, "Hearing aids APP design based on deep learning technology," in *Proc. 11th Int. Symp. Chin. Spoken Lang. Process. (ISCSLP)*, Nov. 2018, pp. 495–496.

[49] N. D. Lane, P. Georgiev, and L. Qendro, "DeepEar: Robust smartphone audio sensing in unconstrained acoustic environments using deep learning," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. (UbiComp)*, 2015, pp. 283–294.

[50] J. M. Martin-Donas, A. M. Gomez, I. Lopez-Espejo, and A. M. Peinado, "Dual-channel DNN-based speech enhancement for smartphones," in *Proc. IEEE 19th Int. Workshop Multimedia Signal Process. (MMSP)*, Oct. 2017, pp. 1–6.

[51] E. Malykh, S. Novoselov, and O. Kudashev, "On residual CNN in text-dependent speaker verification task," in *Proc. Int. Conf. Speech Comput.*, Cham, Switzerland: Springer, 2017, pp. 593–601.

[52] S. Novoselov, O. Kudashev, V. Shchemelinin, I. Kremnev, and G. Lavrentyeva, "Deep CNN based feature extractor for text-prompted speaker recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 5334–5338.

[53] R. Prabhavalkar, O. Alsharif, A. Bruguier, and L. McGraw, "On the compression of recurrent neural networks with an application to LVCSR acoustic modeling for embedded speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 5970–5974.

[54] Z. Lu, V. Sindhwani, and T. N. Sainath, "Learning compact recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 5960–5964.

[55] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 4087–4091.

[56] M. Wang, T. Sirlapu, A. Kwasniewska, M. Szankin, M. Bartscherer, and R. Nicolas, "Speaker recognition using convolutional neural network with minimal training data for smart home solutions," in *Proc. 11th Int. Conf. Hum. Syst. Interact. (HSI)*, Jul. 2018, pp. 139–145.

[57] K. Nan, S. Liu, J. Du, and H. Liu, "Deep model compression for mobile platforms: A survey," *Tsinghua Sci. Technol.*, vol. 24, no. 6, pp. 677–693, Dec. 2019.

[58] C. Chen *et al.*, "Deep learning on computational-resource-limited platforms: A survey," *Mobile Inf. Syst.*, vol. 2020, pp. 1–19, Mar. 2020.

[59] W. Dai and D. Berleant, "Benchmarking contemporary deep learning hardware and frameworks: A survey of qualitative metrics," in *Proc. IEEE 1st Int. Conf. Cognit. Mach. Intell. (CogMI)*, Dec. 2019, pp. 148–155.

[60] Z. Wang, K. Liu, J. Li, Y. Zhu, and Y. Zhang, "Various frameworks and libraries of machine learning and deep learning: A survey," *Arch. Comput. Methods Eng.*, pp. 1–24, Feb. 2019.

[61] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.

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

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

[64] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[65] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[66] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[67] A. K. Gupta, K. Gupta, J. Jadhav, R. V. Deolekar, A. Nerurkar, and S. Deshpande, "Plant disease prediction using deep learning and IoT," in *Proc. 6th Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2019, pp. 902–907.

[68] B. Blanco-Filgueira, D. Garcia-Lesta, M. Fernandez-Sanjurjo, V. M. Brea, and P. Lopez, "Deep learning-based multiple object visual tracking on embedded system for IoT and mobile edge computing applications," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5423–5431, Jun. 2019.

[69] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jul. 2018, pp. 664–669.

[70] T. Na, J. H. Ko, and S. Mukhopadhyay, "Noise-robust and resolution-invariant image classification with pixel-level regularization," in *Proc. Int. Conf. Acoust., Speech Signal Processing,(ICASSP)*, Apr. 2018, pp. 15–20.

[71] M. Song *et al.*, "*In-situ* AI: Towards autonomous and incremental deep learning for IoT systems," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2018, pp. 92–103.

[72] C. M. J. M. Dourado, Jr., S. P. P. da Silva, R. V. M. da Nóbrega, A. C. da S. Barros, P. P. R. Filho, and V. H. C. de Albuquerque, "Deep learning IoT system for online stroke detection in skull computed tomography images," *Comput. Netw.*, vol. 152, pp. 25–39, Apr. 2019.

[73] Y. Wu, X. Zhang, Y. Xiao, and J. Feng, "Attention neural network for water image classification under IoT environment," *Appl. Sci.*, vol. 10, no. 3, p. 909, Jan. 2020.

[74] Y. Chen and Y. Xue, "A deep learning approach to human activity recognition based on single accelerometer," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 1488–1492.

[75] B. Almaslukh, A. Artoli, and J. Al-Muhtadi, "A robust deep learning approach for position-independent smartphone-based human activity recognition," *Sensors*, vol. 18, no. 11, p. 3726, Nov. 2018.

[76] M. M. Hassan, M. Z. Uddin, A. Mohamed, and A. Almogren, "A robust human activity recognition system using smartphone sensors and deep learning," *Future Gener. Comput. Syst.*, vol. 81, pp. 307–313, Apr. 2018.

[77] M. Kim, C. Y. Jeong, and H. C. Shin, "Activity recognition using fully convolutional network from smartphone accelerometer," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2018, pp. 1482–1484.

[78] B. Almaslukh, J. Almuhtadi, and A. Artoli, "An effective deep autoencoder approach for online smartphone-based human activity

recognition," *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 4, pp. 160–165, 2017.

[79] M.-C. Kwon, M. Ju, and S. Choi, "Classification of various daily behaviors using deep learning and smart watch," in *Proc. 9th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2017, pp. 735–740.

[80] Q. Zou, Y. Wang, Q. Wang, Y. Zhao, and Q. Li, "Deep learning-based gait recognition using smartphones in the wild," 2018, *arXiv:1811.00338*.

[81] S. Cortes, A. Solin, and J. Kannala, "Deep learning based speed estimation for constraining strapdown inertial navigation on smartphones," in *Proc. IEEE 28th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Sep. 2018, pp. 1–6.

[82] D. Ravi, C. Wong, B. Lo, and G.-Z. Yang, "Deep learning for human activity recognition: A resource efficient implementation on low-power devices," in *Proc. IEEE 13th Int. Conf. Wearable Implant. Body Sensor Netw. (BSN)*, Jun. 2016, pp. 71–76.

[83] S. Wan, Y. Liang, Y. Zhang, and M. Guizani, "Deep multi-layer perceptron classifier for behavior analysis to estimate Parkinson's disease severity using smartphones," *IEEE Access*, vol. 6, pp. 36825–36833, 2018.

[84] C. Stamate *et al.*, "Deep learning Parkinson's from smartphone data," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2017, pp. 31–40.

[85] A. Shrestha and M. Won, "DeepWalking: Enabling smartphone-based walking speed estimation using deep learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[86] W. Shao, H. Luo, F. Zhao, C. Wang, A. Crivello, and M. Z. Tunio, "DePedo: Anti periodic negative-step movement pedometer with deep convolutional neural networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[87] T. Yoshida, J. Nozaki, K. Urano, K. Hiroi, T. Yonezawa, and N. Kawaguchi, "Gait dependency of smartphone walking speed estimation using deep learning (poster)," in *Proc. 17th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2019, pp. 641–642.

[88] W. Jiang and Z. Yin, "Human activity recognition using wearable sensors by deep convolutional neural networks," in *Proc. 23rd ACM Int. Conf. Multimedia*, Oct. 2015, pp. 1307–1310.

[89] C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Syst. Appl.*, vol. 59, pp. 235–244, Oct. 2016.

[90] J. Yang, K. Cheng, J. Chen, B. Zhou, and Q. Li, "Smartphones based online activity recognition for indoor localization using deep convolutional neural network," in *Proc. Ubiquitous Positioning, Indoor Navigat. Location-Based Services (UPINLBS)*, Mar. 2018, pp. 1–7.

[91] V. Radu, N. D. Lane, S. Bhattacharya, C. Mascolo, M. K. Marina, and F. Kawsar, "Towards multimodal deep learning for activity recognition on mobile devices," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput., Adjunct*, Sep. 2016, pp. 185–188.

[92] P. Vepakomma, D. De, S. K. Das, and S. Bhansali, "A-wristocracy: Deep learning on wrist-worn sensing for recognition of user complex activities," in *Proc. IEEE 12th Int. Conf. Wearable Implant. Body Sensor Netw. (BSN)*, Jun. 2015, pp. 1–6.

[93] P. Sundaramoorthy, G. K. Gudur, M. R. Moorthy, R. N. Bhandari, and V. Vijayaraghavan, "HARNet: Towards on-device incremental learning using deep ensembles on constrained devices," in *Proc. 2nd Int. Workshop Embedded Mobile Deep Learn.*, Jun. 2018, pp. 31–36.

[94] C. Xu, D. Chai, J. He, X. Zhang, and S. Duan, "InnoHAR: A deep neural network for complex human activity recognition," *IEEE Access*, vol. 7, pp. 9893–9902, 2019.

[95] Z. Peng, S. Gao, Z. Li, B. Xiao, and Y. Qian, "Vehicle safety improvement through deep learning and mobile sensing," *IEEE Netw.*, vol. 32, no. 4, pp. 28–33, Jul./Aug. 2018.

[96] X. Liang and G. Wang, "A convolutional neural network for transportation mode detection based on smartphone platform," in *Proc. IEEE 14th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2017, pp. 338–342.

[97] X. Liang, Y. Zhang, G. Wang, and S. Xu, "A deep learning model for transportation mode detection based on smartphone sensing data," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 12, pp. 5223–5235, Dec. 2020.

[98] S.-R. G. Christopoulos, S. Kanarachos, and A. Chroneos, "Learning driver braking behavior using smartphones, neural networks and the sliding correlation coefficient: Road anomaly case study," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 1, pp. 65–74, Jan. 2018.

[99] S.-H. Fang, Y.-X. Fei, Z. Xu, and Y. Tsao, "Learning transportation modes from smartphone sensors based on deep neural network," *IEEE Sensors J.*, vol. 17, no. 18, pp. 6111–6118, Sep. 2017.

[100] H. Zhao, C. Hou, H. Alrobassy, and X. Zeng, "Recognition of transportation state by smartphone sensors using deep bi-LSTM neural network," *J. Comput. Netw. Commun.*, vol. 2019, pp. 1–11, Jan. 2019.

[101] W. Yuan, Y. Jiang, H. Li, and M. Cai, "A lightweight on-device detection method for Android malware," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 9, pp. 5600–5611, Sep. 2021.

[102] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: Automatic framework for Android malware detection using deep learning," *Digit. Invest.*, vol. 24, pp. S48–S59, Mar. 2018.

[103] N. McLaughlin *et al.*, "Deep Android malware detection," in *Proc. 7th ACM Conf. Data Appl. Secur. Privacy*, 2017, pp. 301–308.

[104] R. Feng *et al.*, "MobiDroid: A performance-sensitive malware detection system on mobile platform," in *Proc. 24th Int. Conf. Eng. Complex Comput. Syst. (ICECCS)*, Nov. 2019, pp. 61–70.

[105] A. Gharib and A. Ghorbani, "DNA-Droid: A real-time Android ransomware detection framework," in *Proc. 11th Int. Conf., Netw. Syst. Secur. (NSS)*. Helsinki, Finland: Springer, Aug. 2017, pp. 184–198.

[106] M. Abhijith, B. K. Priya, and N. Ramasubramanian, "Malware detection in Android using machine learning on chip," in *Computing in Engineering and Technology* (Advances in Intelligent Systems and Computing), vol. 1025. Singapore: Springer, 2020, pp. 287–295, doi:10.1007/978-981-32-9515-5_27.

[107] A. Rago, G. Piro, G. Boggia, and P. Dini, "Multi-task learning at the mobile edge: An effective way to combine traffic classification and prediction," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10362–10374, Sep. 2020.

[108] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 445–458, Feb. 2019.

[109] S. Yao, T. Wang, J. Li, and T. Abdelzaher, "Stardust: A deep learning serving system in IoT: Demo abstract," in *Proc. 17th Conf. Embedded Netw. Sensor Syst.*, Nov. 2019, pp. 402–403.

[110] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "DeepThings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2348–2359, Nov. 2018.

[111] C. Gao, S. Braun, I. Kiselev, J. Anumula, T. Delbruck, and S.-C. Liu, "Real-time speech recognition for IoT purpose using a delta recurrent neural network accelerator," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[112] P. Wang, F. Ye, X. Chen, and Y. Qian, "DataNet: Deep learning based encrypted network traffic classification in SDN home gateway," *IEEE Access*, vol. 6, pp. 55380–55391, 2018.

[113] B. Mao *et al.*, "A novel non-supervised deep-learning-based network traffic control method for software defined wireless networks," *IEEE Wireless Commun.*, vol. 25, no. 4, pp. 74–81, Aug. 2018.

[114] F. Tang *et al.*, "On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 154–160, Feb. 2018.

[115] G. Marín, P. Casas, and G. Capdehourat, "RawPower: Deep learning based anomaly detection from raw network traffic measurements," in *Proc. ACM SIGCOMM Conf. Posters Demos*, Aug. 2018, pp. 75–77.

[116] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, 2020.

[117] A. Ignatov *et al.*, "AI benchmark: Running deep neural networks on Android smartphones," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, Sep. 2018, p. 0.

[118] A. Ignatov *et al.*, "AI benchmark: All about deep learning on smartphones in 2019," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 3617–3635.

[119] S. Venticinque, "Benchmarking physical and virtual IoT platforms," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Jun. 2019, pp. 247–252.

[120] H. Kong *et al.*, "EDLAB: A benchmark for edge deep learning accelerators," *IEEE Des. Test*, early access, Jul. 6, 2022, doi: 10.1109/MDAT.2021.3095215.

[121] W. Niu *et al.*, "PatDNN: Achieving real-time DNN execution on mobile devices with pattern-based weight pruning," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2020, pp. 907–922.

[122] X. Cheng, W. Lu, X. Huang, S. Hu, and K. Chen, "HAFLO: GPU-based acceleration for federated logistic regression," 2021, *arXiv:2107.13797*.

[123] J. Qiu *et al.*, "Going deeper with embedded FPGA platform for convolutional neural network," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2016, pp. 26–35.

[124] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2015, pp. 161–170.

[125] F. Akopyan *et al.*, "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[126] S. Rivas-Gomez, A. J. Pena, D. Moloney, E. Laure, and S. Markidis, "Exploring the vision processing unit as co-processor for inference," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2018, pp. 589–598.

[127] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit.*, vol. 44, no. 3, Jun. 2016, pp. 243–254.

[128] T. Chen *et al.*, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *SIGARCH Comput. Archit. News*, vol. 42, no. 1, pp. 269–284, Feb. 2014.

[129] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," 2016, *arXiv:1602.07360*.

[130] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. 14th Eur. Conf. Comput. Vis. (ECCV)*. Amsterdam, The Netherlands: Springer, Oct. 2016, pp. 21–37.

[131] TensorFlow. (2020). *Object Detection*. [Online]. Available: https://www.tensorflow.org/lite/models/object_detection/overview

[132] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[133] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object

detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[134] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7263–7271.

[135] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Esann*, vol. 3, 2013, p. 3.

[136] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explor. Newslett.*, vol. 12, no. 2, pp. 74–82, Mar. 2011.

[137] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.

[138] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proc. 16th Eur. Conf. Cyber Warfare Secur.*, 2017, pp. 361–369.

[139] N. Lane, S. Bhattacharya, A. Mathur, C. Forlivesi, and F. Kawsar, "DXTK: Enabling resource-efficient deep learning on mobile and embedded devices with the DeepX toolkit," in *Proc. 8th EAI Int. Conf. Mobile Comput., Appl. Services*, 2016, pp. 98–107.

[140] J. Wang, B. Cao, P. Yu, L. Sun, W. Bao, and X. Zhu, "Deep learning towards mobile applications," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 1385–1393.

[141] S. Duan, D. Zhang, Y. Wang, L. Li, and Y. Zhang, "JointRec: A deep-learning-based joint cloud video recommendation framework for mobile IoT," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1655–1666, Mar. 2020.

[142] B. Shi, M. Sun, C.-C. Kao, V. Rozgic, S. Matsoukas, and C. Wang, "Compression of acoustic event detection models with low-rank matrix factorization and quantization training," 2019, *arXiv:1905.00855*.

[143] F. Tung and G. Mori, "CLIP-Q: Deep network compression learning by in-parallel pruning-quantization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7873–7882.

[144] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[145] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 742–751.

[146] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani, "N2N learning: Network to network compression via policy gradient reinforcement learning," 2017, *arXiv:1709.06030*.

[147] B. Moons, B. De Brabandere, L. Van Gool, and M. Verhelst, "Energy-efficient ConvNets through approximate computing," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2016, pp. 1–8.

[148] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8612–8620.

[149] I. Markovsky and K. Usevich, *Low Rank Approximation*, vol. 139. Cham, Switzerland: Springer, 2012.

[150] A. Yaguchi, T. Suzuki, S. Nitta, Y. Sakata, and A. Tanizawa, "Decomposable-Net: Scalable low-rank compression for neural networks," 2019, *arXiv:1910.13141*.

[151] W. Shi, Y. Hou, S. Zhou, Z. Niu, Y. Zhang, and L. Geng, "Improving device-edge cooperative inference of deep learning via 2-step pruning," 2019, *arXiv:1903.03472*.

[152] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*, vol. 29, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/d8330f857a17c53d217014ee776bfd50-Paper.pdf

[153] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," 2017, *arXiv:1711.11294*.

[154] R. Ding, T.-W. Chin, Z. Liu, and D. Marculescu, "Regularizing activation distribution for training binarized deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11408–11417.

[155] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proc. ICML*, 2011, pp. 265–272.

[156] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Proc. Workshop Mach. Learn. High-Perform. Comput. Environ.*, 2015, pp. 1–5.

[157] S. S. Latifi Oskouei, H. Golestani, M. Hashemi, and S. Ghiasi, "CNNdroid: GPU-accelerated execution of trained deep convolutional neural networks on android," in *Proc. 24th ACM Int. Conf. Multimedia*, Oct. 2016, pp. 1201–1205.

[158] C. Holmes, D. Mawhirter, Y. He, F. Yan, and B. Wu, "GRNN: Low-latency and scalable RNN inference on GPUs," in *Proc. 14th EuroSys Conf.*, Mar. 2019, pp. 1–16.

[159] P. Hill *et al.*, "DeftNN: Addressing bottlenecks for DNN execution on GPUs via synapse vector elimination and near-compute data fission," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2017, pp. 786–799.

[160] T. Luo *et al.*, "Dadiannao: A neural network supercomputer," *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 73–88, Jan. 2017.

[161] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[162] Y. Ma, Y. Cao, S. Vrudhula, and J.-S. Seo, "Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2017, pp. 45–54.

[163] TensorFlow. (2020). *Tensorflow Hub*. [Online]. Available: https://tfhub.dev/

[164] Keras. (2020). *Keras Applications*. [Online]. Available: https://keras.io/api/applications/

[165] Caffe2. (2020). *Caffe2 Model Zoo*. [Online]. Available: https://caffe2.ai/docs/zoo.html

[166] (2020). *Model Optimization*. [Online]. Available: https://www.tensorflow.org/lite/performance/model_optimization

[167] (2020). *Post-Training Quantization*. [Online]. Available: https://www.tensorflow.org/lite/performance/post_training_quantization

[168] (2020). *Trim Insignificant Weights*. [Online]. Available: https://www.tensorflow.org/model_optimization/guide/pruning

[169] PyTorch. *Quantization*. Accessed: Mar. 1, 2019. [Online]. Available: https://pytorch.org/docs/stable/quantization

[170] I. A. Lab. *Using Distiller to Prune a Pytorch Language Model*. Accessed: Oct. 27, 2020. [Online]. Available: https://github.com/NervanaSystems/distiller/

[171] (2020). *Caffe2 Quantization*. [Online]. Available: https://github.com/pytorch/pytorch/tree/master/caffe2/quantization/serv%er

[172] Microsoft. *Mmdnn*. Accessed: Aug. 13, 2020. [Online]. Available: https://github.com/microsoft/MMdnn

[173] F. Microsoft. *Onnx*. Accessed: Feb. 17, 2021. [Online]. Available: https://onnx.ai/

[174] Z. Xu, F. Yu, J. Xiong, and X. Chen, "Helios: Heterogeneity-aware federated learning with dynamically balanced collaboration," 2019, *arXiv:1912.01684*.

[175] M. Sankupellay and D. Konovalov, "Bird call recognition using deep convolutional neural network, ResNet-50," in *Proc. Acoust.*, vol. 7, 2018, pp. 1–8.

[176] A. Imteaj and M. H. Amini, "FedPARL: Client activity and resource-oriented lightweight federated learning model for resource-constrained heterogeneous IoT environment," *Frontiers Commun. Netw.*, vol. 2, p. 10, Apr. 2021.

[177] A. Hard *et al.*, "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*.

[178] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent IoT applications: A cloud-edge based framework," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 35–44, 2020.

[179] Y. Jiang *et al.*, "Model pruning enables efficient federated learning on edge devices," 2019, *arXiv:1909.12326*.

[180] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14. [Online]. Available: https://openreview.net/forum?id=HJGXzmspb

[181] R. Raju, D. Gope, U. Thakker, and J. Beu, "Understanding the impact of dynamic channel pruning on conditionally parameterized convolutions," in *Proc. 2nd Int. Workshop Challenges Artif. Intell. Mach. Learn. Internet Things*, Nov. 2020, pp. 27–33.

[182] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, and C.-Z. Xu, "Dynamic channel pruning: Feature boosting and suppression," 2018, *arXiv:1810.05331*.

[183] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 409–424.

[184] Z. Yang, S. Hu, and K. Chen, "FPGA-based hardware accelerator of homomorphic encryption for efficient federated learning," 2020, *arXiv:2007.10560*.

[185] J. Mao, X. Chen, K. W. Nixon, C. Krieger, and Y. Chen, "MoDNN: Local distributed mobile computing system for deep neural network," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 1396–1401.

[186] R. Hadidi, J. Cao, M. Woodward, M. S. Ryoo, and H. Kim, "Distributed perception by collaborative robots," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3709–3716, Oct. 2018.

[187] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 595–608, Apr. 2020.

[188] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot-network-based detection of IoT BotNet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Oct. 2018.

[189] B. Roy and H. Cheung, "A deep learning approach for intrusion detection in Internet of Things using bi-directional long short-term memory recurrent neural network," in *Proc. 28th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Nov. 2018, pp. 1–6.

[190] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, Apr. 2016.

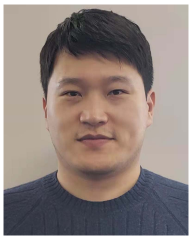
[191] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[192] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar, "Squeezing deep learning into mobile and embedded devices," *IEEE Pervasive Comput.*, vol. 16, no. 3, pp. 82–88, Jul. 2017.

[193] F. Tu, W. Wu, S. Yin, L. Liu, and S. Wei, "RANA: Towards efficient neural acceleration with refresh-optimized embedded DRAM," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 340–352.

[194] D. Marculescu, D. Stamoulis, and E. Cai, "Hardware-aware machine learning: Modeling and optimization," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2018, pp. 1–8.

ABOUT THE AUTHORS



Tianming Zhao (Member, IEEE) received the B.E. degree from the Department of Software Engineering, Chongqing University, Chongqing, China, in 2013, and the M.S. degree from the Department of Computer Science, Binghamton University, Binghamton, NY, USA, in 2016. He is currently working toward the Ph.D. degree at the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA.

He is currently working with Prof. Yan Wang. His research interests include mobile computing and sensing, cybersecurity and privacy, and smart health.

Yucheng Xie received the M.S. degree from the Department of Computer Science, Stevens Institute of Technology, Hoboken, NJ, USA, in 2018. He is currently working toward the Ph.D. degree at the Department of Electrical and Computer Engineering, Purdue School of Engineering and Technology, Indiana University–Purdue University Indianapolis, Indianapolis, IN, USA.

He is currently working with Prof. Xiaonan Guo. His research interests include smart healthcare, mobile sensing and computing, and cybersecurity and privacy.

Yan Wang (Member, IEEE) received the Ph.D. degree in electrical engineering from the Stevens Institute of Technology, Hoboken, NJ, USA, in 2015.

He was with the Department of Computer Science, State University of New York (SUNY), Binghamton, NY, USA. He is currently an Assistant Professor with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA. His research interests include cybersecurity and privacy, the Internet of Things, mobile and pervasive computing, and smart healthcare. His research is supported by the National Science Foundation (NSF). His research has been reported in numerous media outlets, including *MIT Technology Review*, CNN, Fox News Channel, The Wall Street Journal, National Public Radio, and *IEEE Spectrum*.

Dr. Wang was a recipient of the Best Paper Award from IEEE CNS 2018, IEEE SECON 2017, and ACM AsiaCCS 2016. He was the winner of the ACM MobiCom Student Research Competition in 2013.

Jerry Cheng was an Assistant Professor with the Robert Wood Johnson Medical School, Rutgers University, New Brunswick, NJ, USA. He was formerly a Postdoctoral Researcher with the Department of Statistics, Columbia University, New York, NY, USA. He had extensive industrial experiences as a Member of Technical Staff at AT&T Labs, Murray Hill, NJ, USA. He is currently an Assistant Professor of computer science with the New York Institute of Technology, New York. His background is a combination of computer science, statistics, and physics. His work has been reported by many new media, including *MIT Technology Review*, Yahoo News, Digital World, FierceHealthcare, and WTOP Radio. His research interests include big data analytics, statistical learning, Bayesian statistics, and their applications in computer systems and smart healthcare.

Xiaonan Guo (Member, IEEE) received the Ph.D. degree in computer science and engineering from The Hong Kong University of Science and Technology, Hong Kong, in 2013.

He is currently an Assistant Professor with the Department of Computer, Information and Technology, Indiana University–Purdue University Indianapolis (IUPUI), Indianapolis, IN, USA. Prior to joining IUPUI, he was a Research Associate with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ, USA. His research interests include pervasive computing, mobile computing, and cybersecurity and privacy.

Dr. Guo received the Best Paper Award from the ACM Conference on Information, Computer, and Communications Security (ASIACCS) in 2016 and the EAI International Conference on IoT Technologies for HealthCare (EAI Healthy IoT) in 2019.

Bin Hu received the B.Sc. degree in mechanical engineering and the M.Sc. degree in software engineering from Xi'an Jiaotong University, Xi'an, China, in 2000 and 2005, respectively, and the M.Sc. degree from Monmouth University, Long Branch, NJ, USA, in 2018. He is currently working toward the Ph.D. degree in electrical and computer engineering at Rutgers University, New Brunswick, NJ, USA.

From 2006 to 2016, he was an Assistant Professor with the Xi'an University of Posts and Telecommunications, Xi'an. He is currently working with Prof. Yingying Chen. His research interests include mobile computing and sensing, robotics, and machine learning.

Yingying (Jennifer) Chen (Fellow, IEEE) is currently a Professor of electrical and computer engineering and a Peter Cherasia Endowed Faculty Scholar with Rutgers University, New Brunswick, NJ, USA. She is also the Associate Director of the Wireless Information Network Laboratory (WINLAB). She also leads the Data Analysis and Information Security (DAISY) Laboratory. Her background is a combination of computer science, computer engineering, and physics. She had extensive industry experience at Nokia, Murray Hill, NJ, USA. Her research has been reported in numerous media outlets, including *MIT Technology Review*, CNN, Fox News Channel, The Wall Street Journal, National Public Radio, and *IEEE Spectrum*. She has published over 200 journal articles and conference papers. Her research interests include mobile sensing and computing, cybersecurity and privacy, the Internet of Things, and smart healthcare.

Prof. Chen was a recipient of multiple best paper awards from EAI HealthyIoT 2019, IEEE CNS 2018, IEEE SECON 2017, ACM AsiaCCS 2016, IEEE CNS 2014, and ACM MobiCom 2011; the NSF CAREER Award; the Google Faculty Research Award; and the IEEE Region 1 Technological Innovation in Academic Award. She received the NJ Inventors Hall of Fame Innovator Award. She has been serving/served on the Editorial Board of IEEE TRANSACTIONS ON MOBILE COMPUTING (IEEE TMC), IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS (IEEE TWireless), IEEE/ACM TRANSACTIONS ON NETWORKING (IEEE/ACM ToN), and *ACM Transactions on Privacy and Security*.