Hindawi
Mobile Information Systems
Volume 2020, Article ID 8454327, 19 pages
https://doi.org/10.1155/2020/8454327

Review Article

Deep Learning on Computational-Resource-Limited Platforms: A Survey

Chunlei Chen,[1] **Peng Zhang**,[1] **Huixiang Zhang**,[2] **Jiangyan Dai**,[1] **Yugen Yi**,[3] **Huihui Zhang**,[1] **and Yonghui Zhang**[1]

[1]*School of Computer Engineering, Weifang University, Weifang, China*
[2]*School of Cyberspace Security, Northwestern Polytechnical University, Xi'an, China*
[3]*School of Software, Jiangxi Normal University, Nanchang, China*

Correspondence should be addressed to Chunlei Chen; chunlei.chen@wfu.edu.cn

Received 16 August 2019; Revised 28 December 2019; Accepted 1 February 2020; Published 29 February 2020

Guest Editor: Malik Jahan Khan

Nowadays, Internet of Things (IoT) gives rise to a huge amount of data. IoT nodes equipped with smart sensors can immediately extract meaningful knowledge from the data through machine learning technologies. Deep learning (DL) is constantly contributing significant progress in smart sensing due to its dramatic superiorities over traditional machine learning. The promising prospect of wide-range applications puts forwards demands on the ubiquitous deployment of DL under various contexts. As a result, performing DL on mobile or embedded platforms is becoming a common requirement. Nevertheless, a typical DL application can easily exhaust an embedded or mobile device owing to a large amount of multiply and accumulate (MAC) operations and memory access operations. Consequently, it is a challenging task to bridge the gap between deep learning and resource-limited platforms. We summarize typical applications of resource-limited deep learning and point out that deep learning is an indispensable impetus of pervasive computing. Subsequently, we explore the underlying reasons for the high computational overhead of DL through reviewing the fundamental concepts including capacity, generalization, and backpropagation of a neural network. Guided by these concepts, we investigate on principles of representative research works, as well as three types of solutions: algorithmic design, computational optimization, and hardware revolution. In pursuant to these solutions, we identify challenges to be addressed.

1. Introduction

The last decade has witnessed exciting development of deep learning (DL) technologies, which contributes dramatic progress in signal and information processing applications including IoT and smart sensing. A deep neural network (DNN) comprises multiple neuron layers organized in a hierarchical structure. Parameters of every layer can be learned through iterative training. A well-trained DNN can distill useful features from raw data. All training samples are manually labeled. In one layer, input data can be mapped into a low-dimensional space through feature extraction. Subsequently, output features of the current layer are exported into the next layer. Outputs of the last layer imply the learned labels. A DNN can be fine-tuned through minimizing the error between manual labels and learned labels [1].

Deep learning enjoys significant advantages over traditional machine learning [2, 3]. First, deep learning can achieve superior performance when data volume is massive. This means that deep learning can fully benefit from the huge amount of data collected by IoT. Traditional machine learning techniques are preferable when data volume is small. However, the performance prominently degrades when data volume is extremely large. In contrast, deep learning exhibits advantageous scalability with massive data. Second, deep learning relies less on feature engineering. IoT can gather diversified categories of data that are distinct in nature. Manually extracting features of heterogeneous data is a daunting task. Traditional machine learning requires a

domain expert to extract features. The manually identified features expose underlying patterns to algorithms. Nevertheless, deep learning autonomously extract features in a layer-wise manner to represent input samples with a nested hierarchy of features. Every layer defines higher-level features based on lower-level features extracted by the previous layer. Third, deep learning techniques can outperform traditional ones in terms of various smart-sensing-related tasks, such as computer vision, speech recognition, and human behavior understanding.

By contrast with traditional machine learning solutions, deep learning techniques are undergoing rapid development. Applications of deep learning involve information retrieval [4], natural language processing [5], human voice recognition [6], computer vision [7], anomaly detection [8], recommendation systems [9], bioinformatics [10], medicine [11, 12], crop science [13], earth science [14], robotics [15–18], transportation engineering [19], communication technologies [20–22], and system simulation [23, 24].

Deep learning is permeating into diversified aspects of human society, which puts forwards urgent demand on the ubiquitous deployment of DL-powered applications. In other words, deep learning is required to be fit into resource-limited platforms like smartphones or wearable devices. Nevertheless, matching DL and resource-limited platforms is a challenging task. Inferencing with DL is extremely resource-consuming (processor, memory, energy, etc) even though the more resource-consuming training phase can be offloaded onto high-performance-computing-powered mainframes. We investigate on typical resource-limited DL inferencing solutions by categorizing the solutions and discussing open questions. The rest of this paper is organized as follows. Section 2 clarifies impetus of developing resource-limited DL. Representative solutions are discussed in Section 3. Section 4 points out the challenges to be addressed. Section 5 concludes our work.

2. Computational-Resource-Limited Context of Deep Learning

2.1. Application Scenarios. Figure 1 shows typical applications of computational-resource-limited DL in the smart sensing context, including self-driving [25, 26], artificial intelligence APPs of smartphones [27], health/homecare robots [28–31], and intelligent wearable devices [32]. The DNN can be pretrained on remote cloud while the mobile DL platforms communicate with the cloud and perform inference based on local computational and energy resources [33]. All these applications rely on embedded computer with limited onboard resources such as processor, memory, and battery. Two fundamental technologies of such applications are sensor data processing and computer vision.

Recognizing and feeding back to user behavior and surrounding environment are the core functionalities of state-of-the-art Internet-of-Things (IoT) and mobile sensing applications. Nevertheless, raw sensor data are inevitably mixed with noise and uncertainty due to the complicated deployment environment. As a result, distilling precise and meaningful knowledge from raw sensor data is a challenging task. DL is one of the most competitive methods to conquer this challenge [34].

The prevalence of wearable (head-mounted) augmented reality (AR) devices has open a way to a novel class of mobile computer vision applications, including the Microsoft HoloLens [35] and the Google Glass [36]. These applications vary from real-time traffic signal identification for navigation to human recognition for healthcare APPs. All these application scenarios propose the common demand to process continuous video streams in real time. The current leading-edge technology of video stream processing is DL, which handles video streams using a large-scale and pretrained convolutional neural network (CNN) or recurrent neural network (RNN) [37].

2.2. A Perspective of Pervasive Computing. Deep learning can automatically extract features and achieve higher accuracy than traditional artificial intelligence techniques. As a result, deep learning is applicable to a broad range of scenarios. Additionally, open-source development tools like TensorFlow and Caffe are also speeding up progresses in deep learning. Research works on fitting deep learning into resource-limited mobile or embedded platforms will undoubtedly push a huge step forward towards the pervasive deep learning.

Deep learning is currently an indispensable impetus that advances the progress of pervasive computing. As shown in Figure 2, we summarize the development of pervasive computing into three stages. The hardware and software solutions of a former stage are incorporated into the latter stages. In the 1990s, researchers in this area try to facilitate the daily life of humans through Internet-interconnected desktops and mainframes. TCP/IP protocols account for the backbone of networks and the software layer of pervasive applications typically focuses on network organization and data delivery. In the following stage, the mobile Internet provides network access to users at any time and any place. IoT interconnects almost all digital sensors to collect raw data from diversified sources, which results in large data volume and puts forward high demand on the computing power of the data processing platform. Thus, distributed or parallel middleware like Hadoop aggregates the computing power of huge amounts of commodity servers. Additionally, cloud computing provides the aggregated supercomputing power to customers through Web Service. Data transmission between IoT and cloud platforms is further supported by WIFI and 3G/4G. However, applications of this stage mainly adopt traditional machine learning solutions, which cannot achieve constantly advancing performance with the continuous increase of input data volume. Nowadays, the learning and inference accuracy of DNN can efficiently scale with the input data amount. However, high time and memory overheads impede the deployment of DL on resource-limited platforms. Matching deep learning and hardware platforms is an active research area. Software layer solutions mainly focus on simplifying the trained DNN to approximate a full-status DNN. Hardware layer solutions involve embedded GPUs, artificial intelligence chips, or even

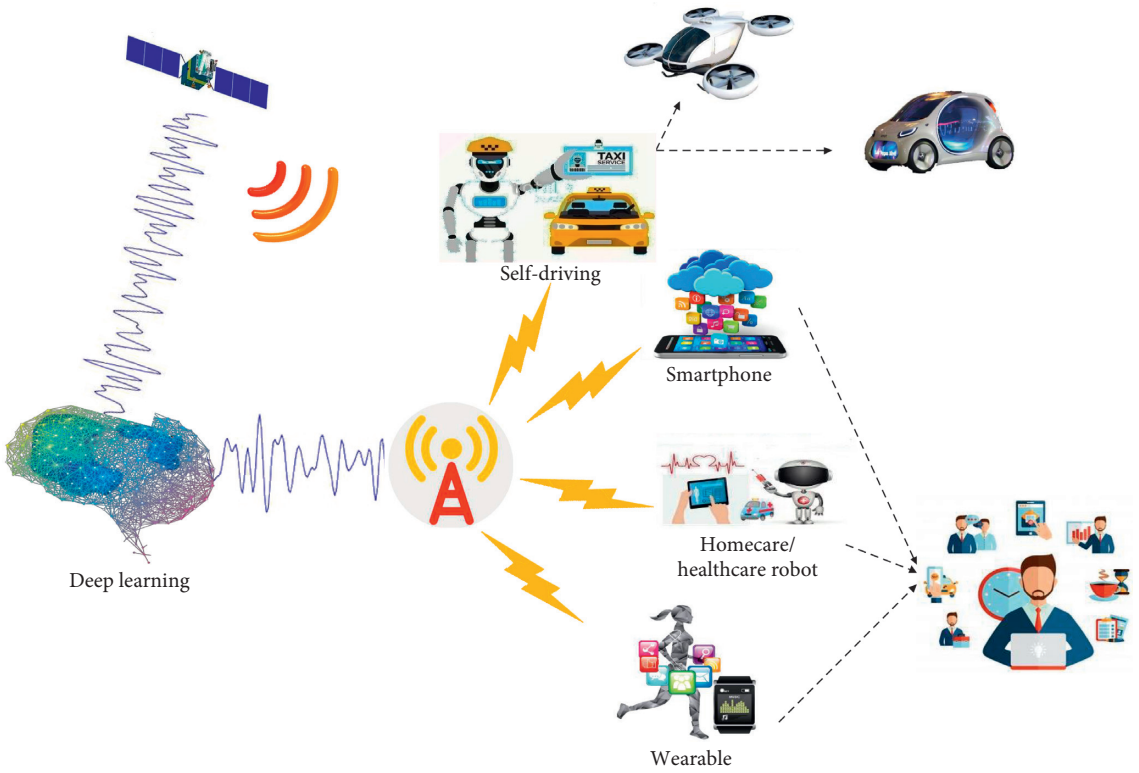

Figure 1: Typical smart-sensing-related application scenarios of resource-limited deep learning.

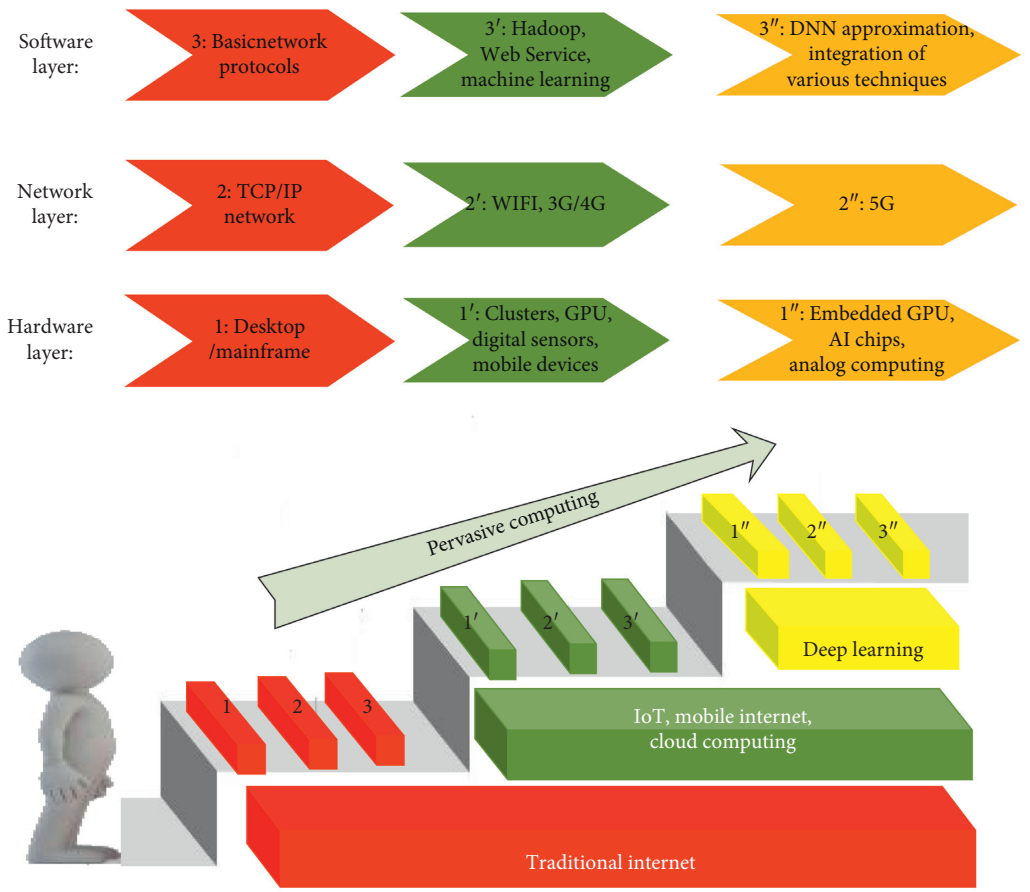

Figure 2: Retrospect of pervasive computing: deep learning is currently one of the dominant impetuses for pervasive computing.

analog computing based on new nonvolatile memory. Additionally, 5G will meet even higher bandwidth requirements.

3. Investigation on Existing Solutions

3.1. Computational Predicament of DNN: A Perspective of Underlying Principles. Classification is a typical application scenario of DNNs. Under this scenario, the target is to establish a mapping from input samples to corresponding labels. The following concepts are the cornerstones to exploit the learning and inference of DNNs: hypothesis space, capacity, stochastic gradient descent, and generalization [38].

Hypothesis space is the set of all functions generated by a neural network. One function is obtained by fitting part of parameters of the neural network and can map homogeneous samples to the same label. Training a neural network is to search the optimal functions in the hypothesis space, which can build mapping relationships specified by the training data (in other words, minimizing the training error). As a result, the size of hypothesis space determines the potential ability of a neural network to find optimal functions.

Capacity of a neural network reflects the size of hypothesis space, as well as the upper bound of ability to fit functions. The optimal functions may be beyond the hypothesis space, if the capacity is not sufficiently large. In this case, the neural network can only search in the limited hypothesis space and find functions that approximate the optimal functions with best efforts. Consequently, underfitting is inevitable.

A trained neural network is expected to correctly predict the label of previously unseen samples. Generalization reflects this kind of ability. Lower generalization error means higher generalization ability. Underfitting during the training phase can result in large generalization error in the inference phase.

Capacity sets the limit of the fitting ability, while generalization can measure the ability of scaling with unknown samples. Another vital issue with neural networks is the mechanism of searching the hypothesis space in the training phase. Conventionally, the searching is manipulated by stochastic gradient descent; searching is always along the direction in which training error drops fastest. The gradients are backpropagated from the deepest layer to the first layer to update weights in a layer-wise manner. Backpropagation converges when the difference of train errors between two successive iterations is smaller than a threshold. However, stochastic gradient descent commonly cannot reach the global optima. Despite that a near-optimal solution is generally sufficient to train a low-error neural network, this method typically requires a long time to converge. Moreover, parameters like step length should be carefully selected to avoid fluctuation of the gradient.

From the perspective of underlying principles, the computational predicament of DNNs is due to the following reasons.

The first is memory overhead. Oversized network is a conventional method to achieve low generalization error. A large capacity does not necessarily result in low generalization error. However, a large hypothesis space raises the upper bound of the generalization ability and thus increases the possibility of reaching a low error, especially when the target functions are not excessively complex.

The second is time and energy overhead. Backpropagation is inherently iterative and time-consuming. The gradient is calculated by minimizing the training error. The training error is a function of weights and other parameters. The huge number of weights leads to a slow convergence speed. Moreover, these weights need to be frequently transmitted between processing units and memory. Consequently, the long-time computation and intensive memory operations raise high demand on the processing ability and energy duration. In addition, values of hyperparameters are conventionally selected through fine-tuning, which multiplies the time overhead.

The third is the curse of dimensionality. High dimensionality of data aggravates the computational resource consumption. DNNs commonly need a large volume of training data to guarantee the generalization ability of the trained network. Higher dimensionality requires denser samples. If A_1 is the number of necessary training data points in the one-dimensional sample space, then the number of training data points is A_1^n in n-dimensional sample space [38]. More training data points of higher dimension inevitably exacerbate overheads of memory, time, and energy.

3.2. Challenges to Be Investigated. Deep learning is currently more art than a science. Neural networks are inherently approximate models and can often be simplified [39].

In spite of the dramatic learning power of deep learning, computational cost has impeded their portability to resource-limited platforms [40]. DL algorithms are facing three kinds of barriers to optimize computational performance. *The first barrier* is the resource-consuming iterative nature of DL training. Moreover, the experiential nature aggravates this kind of iterative cost. Up to now, the success of deep learning mainly relies on empirical designs and experimental evaluations. Theoretical principles are still to be exploited. As a result, optimizing the performance of deep learning requires implementing and executing various possible models within the computational resource constraints to empirically recognize the optimal one [41]. Extracting meaningful knowledge from a single input sample can require enormous MAC operations. The number of MAC operations can reach the magnitude of billion [42]. Additionally, a single deep learning network can contain over a million parameters [43]. As a result, deep learning proposes high demands on processing ability, memory capacity, and energy efficiency. It is a vital issue to optimize deep learning networks by eliminating ineffectual MAC operations and parameters [42]. *The second barrier* is fitting DNNs into diversified modern hardware platforms. Different hardware platforms can be distinct in terms of clock

frequency, memory access latency, intercore communication latency, and parallelism mode. Designer of DL model can be categorized into two different types: data scientist and computer engineer. Data scientists mainly concentrate on optimizing training and inference accuracy through data and neural network techniques. However, they have little or even no concern with computational cost. Efforts to upgrade accuracy do not necessarily result in smaller network size and higher speed. Computer engineers focus on accelerating deep learning based on hardware platforms. They fine-tune or even reform DNNs to match the models to the design requirements for resource-constrained applications. *The third barrier* is lack of dedicated hardware. Traditional general-purpose digital computing hardware such as CPU, GPU, and FPGA neglect some unique characteristics of deep learning. For example, deep learning only involves limited kinds of computational operations. Additionally, deep learning is significantly tolerant to noise and uncertainty. Dedicated hardware may trade off universality for performance [44–48].

Cloud-powered DL has been an active research area. Such solutions can offload heavy computation onto the remote cloud hosts. Such methods assemble data from mobile or embedded devices, transfer the data to cloud, and perform deep learning algorithms (both training and inferencing) on cloud. Users are facing the risk of privacy leakage due to data transmission through computer networks, particularly if the data contain sensitive information. In addition, the reliability of cloud-based deep learning may be affected by network package loss or even network failure. *In this paper, we focus on three issues: first, trade-off between neural network capacity and generalization error using algorithmic design; second, fitting DNN into digital hardware through computational design; and third, next-generation hardware to cope with the computational predicament of DNN.* We categorize the existing solutions into three layers: the algorithmic, computational, and hardware layers.

Figure 3 summarizes typical solutions. A practical method may integrate more than one solutions.

3.3. Algorithmic Design. Algorithmic designs focus on reducing resource consumption through mathematically adjusting or reforming the DNN model and algorithm. Typical simplification techniques include depthwise separable convolution, matrix factorizing, weight matrix sparsification, weight matrix compression, data dimension reduction, and mathematical optimization.

Howard et al. designed a series of neural network models (MobileNets) to facilitate machine vision applications on mobile platforms [49]. MobileNets represent a kind of lightweight deep neural network based on depthwise separable convolutions. The main goal of MobileNets is to construct real-time and low-space-complexity models to satisfy the demands raised by mobile machine vision applications. The contributions of MobileNets are summarized as follows. First, core layers of MobileNets are derived from the depthwise separable convolution. The core concept of the depthwise separable convolution is to factorize a conventional convolution into a depthwise separable convolution layer and a pointwise convolution layer [50]. MobileNets adopt this core concept to reduce the model size, as well as the total number of multiplication and addition operations. Second, pointwise convolutions account for 95% of the total computation while the im2col reordering optimization is unnecessary for pointwise convolutions [51]. Thus, MobileNets avoid massive computation of im2col reordering. Third, since MobileNets generate relatively small models and require comparatively few parameters, conventional anti-overfitting measures are adjusted. For instance, less regularization and data augmentation are used. Additionally, minimal weight decay (L2 regularization) is adopted on the depthwise filter. Fourth, two hyperparameters called width multiplier and resolution multiplier are applied to further shrink the model size.

The core concept of [49] is factorizing a conventional convolution to lower the computation complexity. This factorization does not affect the inference accuracy and thus is a lossless simplification method. However, lossy simplification is necessary if superior simplification effect is demanded. Samraph et al. customize DL network to match FPGA platform [39]. This method simplifies the weight matrix through clustering and encoding. Additionally, matrix-vector multiplication operations are factorized to decrease computational complexity. First, elements of the weight matrix are clustered by k-means into K clusters. Thus, every element is affiliated to a cluster, and the center of every cluster is the mean of its affiliated elements. Consequently, every element in the weight matrix is replaced with the corresponding center. In other words, every weight is approximated with the center of its affiliated cluster. Second, the approximate weights are encoded with a bit width of $\log K$. And all cluster centers form a dictionary vector. As a result, encoding can significantly lower memory overhead. Third, the matrix-vector multiplication can be factorized due to the fact that the encoded matrix has abundant repetitive elements. Therefore, the number of floating-point multiplication operations is dramatically reduced, which means lower computational complexity. In addition to the aforementioned three basic steps, this method faces another problem: replacing weights with cluster centers inevitably induces numerical error to the DL network. This error can affect the inference accuracy. The method of [39] adopts two solutions to handle this error. One is increasing the length of the dictionary vector (in other words, designating a larger K to k-means). The other is to iteratively cluster and retrain the weights. The method of [39] focuses on compressing the already trained weight matrix. By contrast, methods like lasso regularization can sparsify the weight matrix during training [52].

Lane et al. propose a software framework named *DeepX* to reshape the DNN reference model under limited resource constraints [53]. By contrast to the clustering method of [39], *DeepX* uses SVD decomposition and reconstruction error minimization to compress the DNN model. On the first level, they adopt SVD decomposition to reconstruct and approximate the weight matrix of every DNN layer. Thus, *DeepX* dramatically reduces the amount of DNN

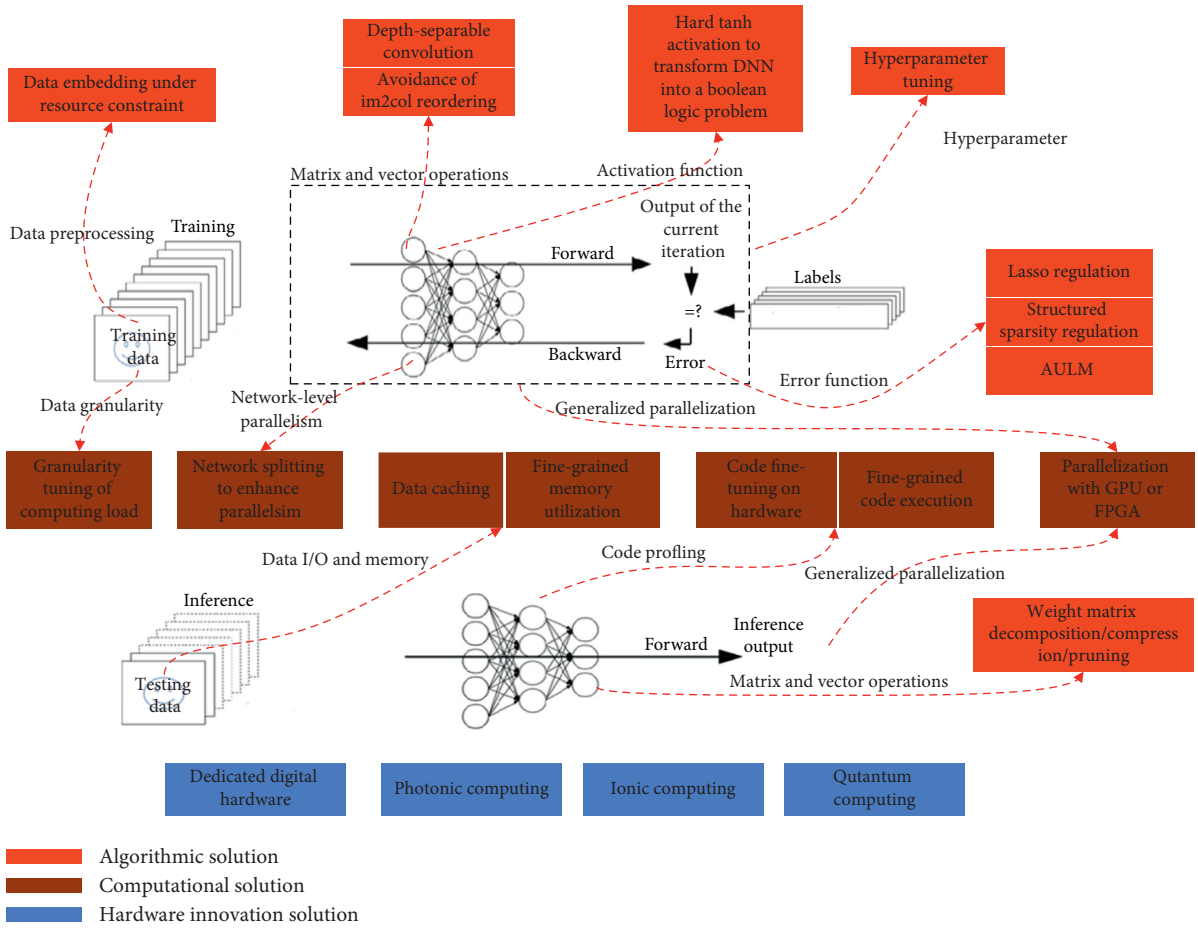

Figure 3: Categorization of existing resource-limited DL solutions: from the perspectives of training and inferencing.

parameters in each layer. Additionally, the accuracy of this approximation is measured and tuned in pursuant to the reconstruction error. As a result, this reconstruction method avoids the predicament of retraining. On the second level, *DeepX* quantizes the computation loads of every neuron and formalizes workload scheduling as a constrained dynamic programming problem. In this manner, computation load can be automatically scheduled onto processors to meet energy and time constraints.

Pruning or compressing an already-trained DNN could result in large approximation error [54–57]. One alternative is to train a sparse DNN. Lin et al. propose a method named structured sparsity regularization (SSR) to achieve weight matrix sparsification during training [58]. They introduce two distinct structured-sparsity regularizers into the object function of matrix weight sparsification. These two regularizers can constrain the intermediate status of DNN filter matrix to be sparse. Subsequently, they adopt an Alternative Updating with Lagrange Multipliers (AULM) scheme to alternatively optimize the sparsification objective function and minimize recognition loss. The SSR method enjoys significantly lower time and memory overhead than state-of-the-art weight matrix pruning methods. Nazemi et al. propose a DNN training method to remove redundant memory access operations. This method utilizes Boolean logic minimization [59]. In the training process, the *sign* function is adopted as the activation. Consequently, activations are confined to binary values. Every layer of the DNN (except the first layer and the last layer) is modeled as a multi-input multioutput Boolean function. In the inference process, outputs of the DNN are obtained through synthesizing a Boolean expression other than computing the dot product of the input and weight. In other words, enormous memory accessing operations are avoided, which removes vast memory access latency and energy consumption.

The aforementioned algorithmic solutions focus on simplifying the DNN model so as to reduce MAC operations and memory consumption. Nevertheless, physical durability, especially energy efficiency, is still a daunting barrier to benefit various practical applications through deep learning. *DeLight* is a low-overhead framework that capacitates efficient training and execution of deep neural networks under low-energy constraints [60]. Authors of [60] restrain the DL network size through energy characterization in pursuant to pertinent physical resources. They design an automatic customization methodology to adaptively fit the DNN into the specific hardware while inducing minimum degradation of learning accuracy. The core concept of *DeLight* is to project data to low-dimensional embeddings (subspaces) in a context-and-resource-aware manner. Consequently,

insights into data samples can be achieved through dramatically less neurons. Moreover, trained models in every embedding are integrated to enhance learning accuracy.

The core concept of *DeLight* is fine-grained energy consumption control based on data dimension reduction. The framework *HyperPower* proposes to bound energy and memory consumption from the point of hyperparameter optimization [41]. This is a hyperparameter optimization framework based on Gaussian process (GP) and Bayesian optimization [61, 62]. This framework denotes test error as a function $f(x)$, where x is a data point in the design space of hyperparameters. Additionally, power and memory overhead is denoted as a function $g(x)$. Subsequently, hyperparameter tuning is formalized as an optimization problem: minimizing $f(x)$ under the constraint that $g(x)$ is lower than a threshold. Minimizing $f(x)$ is costly due to the fact that $f(x)$ has no close form. Consequently, *HyperPower* adopts GP to approximate distributions of $f(x)$. Moreover, the framework leverages Bayesian optimization to iteratively select optimal hyperparameters and update distribution of $f(x)$. $f(x)$ is assumed to obey Gaussian distribution. Let y denote the observations of $f(x)$. At the very beginning, an initial approximation of $f(x)$ can be resolved as $p_M(y \mid x)$ based on the assumption and a set of known (x, y) values (Gaussian process regression). Every iteration includes the following operations. The primary task is to select an optimal value of x from the design space to refine $p_M(y \mid x)$. And the selected x should push the $f(x)$ value along a direction of decrease. This value of x is identified through maximizing an expectation-improvement-based acquisition function. In addition, the acquisition function incorporates the constraint using an indicator function. The indicator function equals to one if the constraint is satisfied and zero if not. Second, the neural network is configured in accordance with the new design parameter (the newly identified x) and trained to obtain the test error (a new value of y). Third, the mean and covariance are updated using the new (x, y), and thus, $p_M(y \mid x)$ is updated to $p_M(y)$.

3.4. Computational Optimization. Computational optimization relies on reengineering the algorithm implementation in accordance with a specific hardware architecture. Some conventional optimization techniques are code parallelizing, fine-tuning of parallel code, data caching, and fine-grained memory utilization.

Huynh et al. developed a tool *DeepMon* for continuous vision applications based on commodity mobile GPUs [37]. Large deep neural networks (DNNs) powered by commodity mobile GPU commonly cannot achieve strict real-time performance due to limited computational resources. However, the frame rate can be low (one to two frames per second) under some use cases, such as speaker recognition and elder nursing care. These application scenarios put forward comparatively low demands on real-time performance. *DeepMon* implements large DNNs for such applications based on commodity mobile GPUs and achieves near real-time performance. In the aforementioned applications, first-person-view images are not apt to exhibit significant changes during a short time span. *DeepMon* divides each frame of image into equal-size blocks. *DeepMon* cached the intermediate results of each block when calculating the convolution of one frame. Subsequently, similar blocks are identified between this frame and the next frame. Consequently, the cached results can be directly utilized to calculate convolution of the next frame. Additionally, cached results expire after a certain time period. Similarity between two images is identified based on color distribution histogram and chi-square distance metric. In addition to this caching mechanism, *DeepMon* leverages Tucker-2 decomposition convolution layers [63] to factorize a traditional convolution layer into several small convolution layers. As a result, computation cost of convolution is reduced. Finally, *DeepMon* tunes GPU codes on various mainstream commodity mobile GPUs. Tuned and optimized GPU codes are encapsulated into separate kernels for each GPU model. As a result, *DeepMon* can adaptively adopt appropriate kernels at runtime so as to fit into a specific GPU with best efforts.

The main idea of *DeepMon* is caching the intermediate result to eliminate redundant computation. Another typical technique is GPGPU acceleration. Cao et al. proposed a GPGPU-powered RNN model that executes locally on mobile devices [64]. Recurrent neural network (RNN) can find wide applications such as speech recognition and robot chatting. Traditional mobile applications of RNN generally offload main computation onto the cloud. However, the cloud-based implementation induces security and efficiency issues. Cao et al. pointed out that existing GPGPU-accelerated methods for convolutional neural network (CNN) cannot directly be transplanted to mobile-device-based RNN. On the one hand, RNN inherently contains many sequential operations, which constrains the parallelism of RNN. On the other hand, existing GPGPU-powered RNN methods are specially designed for desktop GPGPUs. Such methods can not directly fit into mobile GPGPUs due to the fact that the mobile GPGPU possesses significantly less memory capacity and processing cores. In a RNN, the inevitable dependencies between adjacent cells dramatically increase the difficulty in exploiting parallelism among cells. Nevertheless, operations within a cell still exhibit considerable parallelism. In the work of [64], computation of the cell is factorized in fine granularity and elegantly fits into the mobile GPGPU.

The adaptive platform DL framework *Deep*^{3} still adopts the idea of GPGPU-powered computing. However, *Deep*^{3} exploit parallelism from three levels: data, network, and hardware. The ultimate goal of *Deep*^{3} is to bridge the gap between data science perspective design of deep learning and computer engineering perspective optimization of deep learning. First is hardware parallelism. *Deep*^{3} extracts basic operations (layers) of a deep learning network, including convolution, maximum pooling, mean pooling, matrix multiplication, and nonlinearities. Optimized implementation of a basic operation can be dramatically distinct with regard to the hardware platform. For example, by altering the dimensionality of matrices, we can observe that matrix multiplication is computation-intensive or data-intensive on

a specific platform. *Deep*^{3} employs subroutines to perform hardware profiling. Each subroutine runs a specific operation with varying sizes on different platforms, separately. In this manner, *Deep*^{3} recognizes the optimal size of a specific operation regarding a target platform. These optimal sizes are vital instructions to split an entire deep learning network into subnetworks, which adapt the computational, memory, and bandwidth resources of the target platform. Second is network parallelism. *Deep*^{3} breaks down the entire deep learning network into overlapped subnetworks using a depth-first method. Each subnetwork has the same depth as the original network with significantly fewer edges. Every subnetwork can be independently updated, and such local updates are periodically collected by a parameter coordinator to optimize the entire network. Third is data parallelism. *Deep*^{3} decomposes the high-dimensional input data into several low-dimensional subspaces through dictionary learning. Dictionary learning can be efficiently performed by machine learning algorithms like spectral clustering [65–67]. Subsequently, each subnetwork is dedicated to handling a specific subspace and different subspaces are processed in parallel.

Wu et al. exploit mobile deep learning in the joint perspective of software-and-hardware architecture. They propose a platform named *DeepShark* to capacitate commercial-off-the-shelf (COTS) mobile devices with the capability of adaptive resource scheduling [68]. Methods like *DeepX* try to compress the deep model. By contrast, *DeepShark* seeks trade-off between response speed and memory consumption. It splits a pretrained DNN into code blocks and incrementally runs the blocks on system-on-chip (SoC) to accomplish inference. Consequently, *DeepShark* only needs to load currently required data from external storage into memory rather than hold entire data in memory throughout the execution period. Thus, *DeepShark* remarkably lowers memory consumption. In addition, *DeepShark* induces no accuracy loss due to the absence of model compression or approximation. Moreover, privacy risks are avoided due to the fact that all user-relevant data are processed locally. Eventually, *DeepShark* is transparent to deep learning developers. It overloads default system functions of TensorFlow and Caffe. Developers can invoke *DeepShark* APIs in the same way as calling TensorFlow or Caffe APIs. By contrast, the work of [59] eliminates redundant memory operations in an algorithmic manner.

3.5. Hardware Revolution. Haensch et al. point out that the aspiration to apply DL to all fields of daily life is an inheritage of pervasive computing. However, academia and industry are facing challenging barriers to scale DL to fit DL into pervasive applications [69]. Overhead is a vital problem regarding pervasive application of DL, where overhead refers to time and computational resources required to construct, train, and run the model. Prior-art research works show that GPUs take a step further towards pervasive DL, whereas it is confirmed that customized hardware dedicated to DL can outperform general-purpose GPUs.

Han et al. design a dedicated processor for DNN-based real-time object tracking [70]. This processor achieves low power consumption through a DNN-specific processor architecture and a specialized algorithm. However, this dedicated processor still relies on digital computing.

A DL network only requires limited kinds of mathematical operations (for example, matrix multiplication). And such operations frequently reoccur in model training or inference. These two characteristics enable efficient execution of DL algorithms on not only GPUs but also analog computing circuits. Additionally, DL algorithms are highly tolerant to noise and uncertainty, which opens a way to trade numerical precision for algorithmic accuracy. Analog computing discussed by Haensch et al. [69] is an extension of in-memory computing. Prior-art nonvolatile memory materials cannot efficiently accommodate analog in-memory computing. Reengineering memory materials is a challenging task. A new generation of DL accelerating hardware has entered the vision of academia and industry. This kind of hardware trades versatility for low overhead. Nevertheless, complexity of constructing and training DL models is beyond the capacity of any single kind of hardware. As a result, researchers need to consider the solution in a systematic perspective and aggregate several kinds of accelerators into a perfect system. Vitality of new accelerators heavily depends on this issue. Moreover, Haensch et al. declare that analog accelerators will not completely replace the digital ones. Both digital and analog accelerators should be continuously developed to the maximum possible extent. The analog accelerators should be capable of seamless integration into digital ones.

Analog computing can be implemented based on electrochemical reactions. Such a mechanism has been investigated to establish hardware foundations for DL-related problems. For example, neuromorphic computing can circumvent immanent performance bottlenecks of traditional computing via parallel processing and crossbar-memory-enabled data accessing. Fuller et al. link a redox transistor to a conductive-bridge memory (CBM) and thus establish an ionic floating-gate memory (IFG) array [71]. The working life of redox transistors can reach up to over one billion "read-write" operations. Additionally, data access frequencies can achieve more than one megahertz. This IFG-based neuromorphic system shows that in-memory learning and inference can efficiently perform based on low-voltage electrochemical systems. The adaptive electrical features of IFG can hopefully pioneer neuromorphic computers that can significantly outperform conventional digital computers in power efficiency. Such neuromorphic analog computers could adjust deep learning to power-limited context, or even capacitate persistent lifelong learning of a product. Another electrochemistry-based hardware prototype is proposed in [72]. Tsushiya et al. design a solid-state ionic device to address decision-making issues like the multiarmed bandit problem (MBPs). This device opens a way to achieve decision-making through motion of ions, which could contribute to mobile artificial chips and find various applications including deep learning.

In addition to analog computing, photonic (or optical) computing is also a promising hardware solution. Currently, mainstream photonic computers replace components of electric digital computers with photonic equivalents, which can achieve higher speed and bandwidth. Some pioneering research works have adopted photonic computing to support DL-related computations. Rios et al. achieve all-photonic in-memory computations through combining integrated optics with collocated data storage and processing [73]. They fabricate nonvolatile memory using the phase-change material $Ge_2Sb_2Te_5$ and perform direct scalar and matrix-vector multiplications based on this nonvolatile photonic memory. The computation results are represented by the output pulses. This photonic computing system offers a promising shift towards high-speed and large bandwidth on-chip photonic computing, which circumvents electro-optical conversions. Such a system could be the cornerstone of the purely photonic computers. Feldmann et al. point out that conventional computing architectures differentiate real neural tissue by physically separating the functionalities of data memory and processing [74]. This separated design places a daunting barrier to achieving high-speed and power-efficient computing systems like human brains. A promising solution to conquer this barrier is to elaborate novel hardware to simulate neurons and synapses of human brains. Consequently, they investigate on wavelength division multiplexing techniques to implement a photonic neural network based on a scalable circuit, which can mimic the neurosynaptic system in an all-optical manner. This circuit maintains the intrinsic high-speed and large bandwidth characteristics of an optical system and capacitates efficient execution of machine learning algorithms.

Quantum computing is another prospective solution to support DL. Gao et al. adopt a quantum generative model to design quantum algorithm of machine learning. This model enjoys superior ability of representing probability distributions over conventional generative models. In addition, the model can achieve a speedup of exponential magnitude at least in some application scenarios that a quantum computer cannot be fully simulated through conventional digital computing paradigm. The work of [75] opens a way to quantum machine learning and demonstrates a dramatic instance where a quantum algorithm of both theoretical and practical values can reach exponentially higher performance over conventional algorithms.

Novel hardware paradigms like ionic memory, photonic computing, and quantum computing could set indispensable stages for resource-limited deep learning. Despite that these hardware evolutions may be initially motivated by facilitating deep learning applications, the next-generation hardware could find much broader applications in future.

3.6. Discussion. Table 1 summarizes representative works in the perspective of underlying principles that account for the computational predicament of DNNs. Existing research works commonly aim at dealing with one or more of the causes of the computational predicament.

The first is memory overhead induced by oversized network. Earlier algorithmic solutions tend to compress or prune the weight matrix of a pretrained DNN. Compressing or pruning is a trade-off between the capacity (or generalization ability) and memory efficiency. However, directly modifying a pretrained network inevitably results in unexceptable error. Despite that retraining is a choice, it will induce remarkable extra time overhead.

As a result, recent algorithmic solutions propose to achieve a sparse network through training. The core idea is to elaborately select a regularization item for the error function, which forces the network to form sparse weight matrices yet at little or even no loss in generalization ability. In addition to algorithmic solutions, digital computers can also capacitate large pretrained networks in the inference phase through fine-grained utilization of memory.

The second is time or energy overhead induced by backpropagation, memory operations, and hyperparameter tuning. From the point of algorithmic view, dramatic redundant computation can be eliminated, especially in matrix-matrix or matrix-vector multiplications. In this manner, time overhead as well as energy consumption is reduced. Time efficiency can also be promoted by reusing intermediate results of convolution, parallelization on digital processors, and code fine-tuning on digital processors. Unlike overhead caused by arithmetic processing, time consumption induced by memory operations is difficult to handle. The reason is that traditional digital computers adopt von Neumann architecture and thus have independent processing and memory units. Due to the statistical and approximate nature of DNNs, Boolean logic minimization can contribute to the reduction of memory operations, as well as energy consumption. This solution achieves efficient performance in handwritten digital recognition. However, it confines the activation functions to be *sign* functions, which limits the generalization ability. Regarding energy-related hyperparameter tuning, mathematical methods like Gaussian process can point out a more efficient searching path in the parameter space, other than merely rely on human experience or even random searching.

Energy consumption is mainly caused by arithmetic processing and memory operations. Consequently, the latter two are key problems. Regarding time overhead, most existing solutions focus on periphery issues like redundant computations. However, the problem roots in stochastic gradient descent. The training time will drop dramatically if we could fabricate an improved gradient that can lead to convergence more rapidly. With regard to memory operation overhead, it is an inherent problem of the von Neumann architecture. Resolving this problem requires new computing paradigms like in-memory computing.

The third is the curse of dimension. Conventional solutions like weight matrix decomposition and data embedding can reduce the feature dimension. As far as we know, there are limited research works of feature dimension reduction in the computational-resource-limited context. Relevant topics are to be investigated.

It should be noted that the above discussed aspects are not isolated to each other. A systematic view may imply a

Table 1: Representative research works in the perspective of underlying principles.

	Representative research works	Techniques
Memory overhead induced by oversized network	[39]	Weight matrix compression of a pretrained network through clustering: merging similar functions in the hypothesis space
	[56]	Weight pruning of a pretrained network: removing the weights that contribute little to fitting functions in the hypothesis space
	[39, 58]	Sparse training: lasso regularization, structured sparsity regularization
	[68]	Computational optimization on digital computers: fine-grained utilization of memory
Time or energy overhead induced by backpropagation, memory operations, and hyperparameter tuning	[37, 39, 49]	Algorithmic design to avoid computation redundancy: depth separable convolution, avoidance of im2col reordering, factorized matrix-vector multiplication based on SVD and Tucker-2
	[37]	Caching of digital computers: reuse intermediate results of convolution to avoid redundant computation
	[39, 40]	Parallelization on digital processors: FPGA, GPGPU
	[37, 40, 53]	Full utilization of digital processors: profiling and fine-tuning of CPU or GPGPU codes
	[59]	Avoidance of frequent memory operations through Boolean logic minimization
	[41]	Hyperparameter tuning using Gaussian process
Curse of dimension	[53]	SVD decomposition of the weight matrix
	[60]	Data embedding

more efficient solution. For example, a pretrained sparser network undoubtedly demands less inference time than a denser network. Another instance, reading/writing weights will induce less time and energy consumption if the weight matrix is sparser. Table 1 does not cover innovative computing paradigms like analog computing and quantum computing. We will discuss such computing paradigms in more detail later.

Table 2 provides more details on the representative research works. Three categories of solutions are all under rapid development. The overall motivation is to apply DL to mobile/embedded context efficiently. Algorithmic solutions are at the core position due to the fact that they directly cope with business logic of real applications and aim to reduce time and memory complexity on the mathematical logic layer. Existing solutions mainly focus on simplifying matrix-and-vector operations, data/network embedding, hyperparameter tuning, and sparsification through regularization. Further research is still needed to explore reducing computational overhead through activation function.

In addition to the mathematical logic layer, traditional general-purpose digital hardware bridges the gap between mathematical algorithms and real applications. To the best of our knowledge, most practical mobile/embedded DL-based applications are based on traditional hardware. In this case, classical computational optimization methods can be adopted to fully utilize computational resources, including data caching, parallelization, and code fine-tuning. However, many existing DNNs are designed by AI experts, who place little or even no concern on the adaptiveness of DNNs to hardware. As a result, the DNNs may need some reshaping to efficiently fit into a specific hardware device. In view of this, we expect that researchers can design DNNs in a joint view of both AI experts and computer engineers.

Currently, representative computational performance metrics include memory overhead, memory access latency, parallelism (full utilization of processors), and power consumption. However, some topics still remain to be investigated. For instance, *DeepShark* uses external storage as the cache to support fine-grained memory utilization. Power consumption caused by data I/O is to be discussed. In addition, the balance between cache size and cache hit rate is also an interesting topic. Table 3 shows the datasets that were used to evaluate a DNN in pursuant to more than one performance metrics. These datasets and relevant algorithms are favourable choices to serve as benchmarks.

Nevertheless, traditional general-purpose digital hardware may be still inefficient under certain scenarios. Consequently, DL-dedicated digital hardware is becoming increasingly popular, whereas the computational performances of digital hardware are facing bottleneck due to physical constraints. Next-generation computing technologies such as quantum computing are promising solutions to conquer such constraints. Next-generation computing technologies will undoubtedly boost the progress of deep learning even if they are now in their infancies.

4. Challenges to Be Addressed

Despite the promising prospect of existing solutions, we are still facing some considerable challenges to be addressed.

4.1. Fundamental Support for Hardware Revolution. Analog computing is a promising technology to facilitate DL due to the fact that DL is tolerant to numerical errors.

Table 2: Details of representative research works.

	Date	Name (ref. no.)	Resource	Representative method	Architecture of NN or topic of machine learning	Application scenario	Dataset
Algorithmic	2016-4-11	*DeepX*[53]	Memory capacity, power	Inference phase: SVD decomposition-based weight matrix compression, fine-grained task scheduling to processors	AlexNet [76], 2-hidden layer DNN for SpeakerID, SVHN CNN, 2-hidden layer DNN for Audio Scene	Recognition of objects, human voice, audio environment	ImageNet [76], Speaker Verification Spoofing, and Countermeasures Challenge Dataset [77], SVHN dataset [78], Audio Scene dataset [79]
	2016-8-8	*DeLight*[60]	Power	Training phase: data projection under energy constraint	4-Layer DNN	Imaging, smart sensing, speech recognition	Hyperspectral Remote Sensing Scenes [80], UCI Daily and Sports Activities [81], UCI ISOLET [82]
	2017-4-17	*MobileNets*[49]	Memory capacity	Training phase: depthwise separable convolution, avoidance of im2col reordering, hyperparameter tuning	A 28-layer convolution neural net, PlatNet [87, 88], FaceNet [89, 90]	Large-scale geolocation, fine-grained image recognition, face recognition, object detection	ImageNet, Im2GPS [83], Stanford Dogs [84], YFCC100M [85], COCO [86]
	2017-4-30	[39]	Memory capacity	Inference phase: weight encoding, weight sharing, factorization of vector-matrix multiplication	2-Hidden layer DNN	Speech recognition, indoor localization, human activity recognition, handwritten digital recognition	UCI ISOLET, UCI UJIIndoorLoc [87], UCI Daily and Sports Activities, MNIST [88]
	2018-3-19	*HyperPower*[41]	Power	Training phase: hyperparameter tuning, GP-Bayesian optimization	Variants of AlexNet for MNIST and CIFAR-10	Handwritten digital recognition, image classification	MNIST,CIFAR-10 [89]
	2019-1-21	[59]	Memory access latency, power	Training phase: transform the DNN realization problem into a Boolean logic optimization problem, Boolean logic minimization	Multiple layer perception [92], CNN	Handwritten digital recognition	MNIST
	2019-2-28	[52]	Memory capacity	Training phase: group lasso regularization, intergroup lasso regularization	Fully convolutional network with 7 convolution layer initialized with pretrained VGG16	Face recognition	LFW face dataset [93]
	2019-4-12	[58]	Memory capacity	Training phase: structured sparsity regularization, Alternative Updating with Lagrange Multipliers (AULM)	LeNet [94], AlexNet, VGG-16 [95], ResNet-50 [96], GoogLeNet [97]	Handwritten digital recognition, image classification	MNIST, ImageNet

Table 2: Continued.

	Date	Name (ref. no.)	Resource	Representative method	Architecture of NN or topic of machine learning	Application scenario	Dataset
Computational	2017-6-18	*Deep*^{3}[40]	Processor	Training and inference phases: enhancing parallelism through computing load granularity altering, network splitting through depth-first traversal methodology, data dimension reduction using dictionary learning, parallelizing with GPU	Establish an universal framework for fitting DL network into specific hardware, AlexNet was used as an example	Imaging, smart sensing, speech recognition	Hyperspectral Remote Sensing Scenes, UCI Daily and Sports Activities, UCI ISOLET
	2017-6-19	*DeepMon*[37]	Processor, power	Inference phase: data caching, hardware-specific code fine-tuning, Tucker-2 matrix decomposition	VGG-Verydeep-16 [95], YOLO [98]	Continuous vision application	ILSVRC2012 train dataset [99], Pascal VOC 2007 train dataset [100], UCF101 dataset [101], LENA dataset [102]
	2017-6-23	*MobiRNN*[64]	Processor	Inference phase: fine granularity code execution, parallelizing with GPU	LSTM model [103]	Smart sensing	Mobile phone sensor dataset [104]
	2019-2-1	*deepshark*[68]	Memory capacity	Inference phase: fine-grained memory utilization	VGG, CaffeNet [105], GoogLeNet, AlexNet	Imaging	ILSVRC2012
	2018-10-4	[70]	Computing power, power	Training and inference phases: a unified core architecture, binary feedback alignment (BFA), dynamic fixed-point-based run-length compression (RLC), dropout controller	MDNet [106]	Real-time object tracking	Object tracking benchmark (OTB) dataset [107]
Hardware	2018-9-7	[72]	Computing power	Adopt voltage-charge relationship of electrochemical cells to achieve forgetting parameters, describe the decision-making problem using motion of ions	Multiarmed bandit problems (MBPs)	Reinforcement learning	—
	2018-12-7	[75]	Computing power	Quantum computing, model the correlation in data with underlying probability amplitudes of a many-body entangled state	Generative model	Generative model	—
	2019-2-15	[73]	Computing power	Electrochemical cells	Matrix-vector multiplications based on nonvolatile photonic memory	Basic arithmetic operations for machine learning or AI algorithms	—
	2019-5-9	[74]	Processing power, power	Separating the functionalities of data memory and processing, mimic the neurosynaptic system in an all-optical manner	Neural network consisting of four neurons and sixty synapses (and 140 optical elements in total)	Letter recognition	—

Table 3: Datasets relevant to more than one performance metrics.

	Performance metric	Relevant research work
UCI Daily and Sports Activities, UCI ISOLET	Processor utilization rate	[40]
	Memory overhead	[39]
	Power consumption	[60]
ILSVRC2012	Processor utilization rate	[37]
	Power consumption	[37]
	Memory overhead	[68]
MNIST	Memory overhead	[39, 59]
	Memory access latency	[59]
	Power consumption	[41]
Hyperspectral Remote Sensing Scenes	Processor utilization rate	[40]
	Power consumption	[60]

However, analog computing is a kind of in-memory computing and raises the demand for novel nonvolatile memory material. Analog-computing-powered DL calls for long-term joint efforts of computer scientists and material scientists.

Compared to the other innovative types of hardware, analog computing is temporarily taking the leading position. The analog array technology has been successfully applied to DNN to processing common datasets [108], while other innovative hardware technologies such as photonic computing and quantum computing are still to be applied to DNN [73, 75]. Superiority of the analog array lies in the fact that it adopts analog circuit to compute matrix-vector multiplication with constant time overhead irrelevant to size of the matrix. However, it is a predicament to straightly map convolutional neural network onto conventional analog arrays due to the fact that kernel matrices are commonly small and the constant-time multiplication operation has to be iterated for many times in a sequential manner. Rasch et al. parallelize the training through duplicating the kernel matrix of a convolution layer on different analog arrays and stochastically dispatching parts of the computation onto the arrays. As a result, the speedup ratio is proportional to the amount of kernel matrices per layer [106].

In addition to the high speed-up ratio, another advantage of analog computing is the splitting of processing and memory. Under a traditional von Neumann architecture, processing units and memory are separate. Data transmission between processing units and memory can consume orders of magnitude more energy than conventional arithmetic operations. In addition, a typical deep learning application routinely demands enormous data transmission operations, which raises dramatically higher energy consumption than that of computation. One promising solution is collocating processing units and memory using phase-change memory [109].

Despite that analog computing hardware has exhibited promising potential to outperform traditional von Neumann architecture hardware like GPUs, most existing research works focus on the functionality of such analog hardware. Efficiency and reliability issues like stability and durability are yet to be investigated before moving out of the lab to real applications [110].

4.2. More Efficient Algorithmic Solutions. Some algorithmic solutions like weight matrix compression and weight matrix decomposition are approximating the original pretrained neural network with a simplified one. Nevertheless, empirical nature of DL hinders solving an exact theoretical upper bound of approximation error. The absence of this upper bound makes it difficult to prove the robustness of such approximations. Additionally, due to the lack of theoretical principles, many algorithmic techniques require iterative tuning and running the model to select the optimal one. Nevertheless, the design space of model parameters is large. As a result, implementing such algorithmic techniques to large-scale real applications may be a daunting task, especially when we need to deal with hyperparameters within large ranges.

Posttraining simplification of the DNN may result in large error. Moreover, a large number of parameters hinder the stochastic gradient descent to achieve a near-optimal solution. Sparse training is a promising method to cope with these two problems.

Achieving high capacity of a deep neural network is a conventional solution to guarantee low generalization error. However, most deep neural networks obtain high capacity through harnessing a large number of weights, which means dense connections between consecutive layers. This explains the reason that many existing deep neural networks adopt fully connected layers. Nevertheless, real biological scale-free neural networks can significantly outperform state-of-the-art deep learning networks yet with sparse connections. Inspired by this observation, Mocanu et al. construct a sparse scale-free network topology with two consecutive layers [111]. This topology substitutes sparse layers for fully connected layers before training. Their sparse evolutionary training method quadratically decreases the amount of parameters, inducing no loss in accuracy. This sparse training method opens a way to lower the barrier to fitting deep learning into traditional hardware.

Based on the method of [111], Liu et al. train a sparse MLP (multiple layer perception) model with a million neurons to classify microarray genes [110]. This MLP model can be trained within the time of 101 seconds magnitude and achieve lower generalization error than traditional models

(dataset: Leukemia, dimension: 26, 1397 training data samples and 699 testing data samples).

The method of [111] mainly focuses on building a novel network topology, yet still adopts conventional stochastic gradient descent to train the model [111]. Dettmers et al. harness exponentially smoothed gradients to recognize layers and weights that efficiently decrease the training error of a sparse model. As a result, the model can converge significantly faster. In addition, the trained network is insensitive to hyperparameters.

In recent years, a rapidly increasing number of research works are investigating on sparse training of DNN [112–116]. These research works typically concentrate on sparse training of several types of DNN. In view of the diversity and complexity of DNNs, it is a highly valuable yet challenging job to exploit sparse training for various types of DNNs under specific application requirements.

4.3. Systematic Integration. As discussed in Section 2, the ultimate goal of resource-limited DL is ubiquitous deployment of DL. Diversified applications can put forward various requirements on ubiquitous DL. As a result, we need to systematically integrate various types of solutions.

Next-generation computing hardware should seamlessly collaborate with traditional digital hardware, with the ultimate target of accommodating the tachytely evolving DNNs.

Gil and Green argue that the future computing hardware is based on intersections of three aspects: mathematics and information, neuron-inspired biology and information, and physics and information. These intersections give rise to the concepts of digital computing, neural computing, and quantum computing, respectively. Gil and Green denote the three concepts as bit, neuron, and qubit, respectively. As shown in Figure 4, the next-generation AI-enabled computing system requires integration of the three [117]. In this figure, we adopt quantum computing (qubits) to represent future computing paradigms. Novel computing paradigms like analog computing should be also taken into consideration. We discuss this integration in detail as follows.

4.3.1. Digital Computing. The advantage of digital computing lies in its stable binary nature. With the same binary input, a digital computing system should always generate the same output. This nature is the cornerstone of building robust and stable systems for data storage and processing. Classical digital computing is still an efficient solution to not only mathematical and logical operations but also persistent data storage. In the future computing system, digital computing will still occupy an indispensable position due to its robust and reliable nature.

4.3.2. Neuron Computing. Despite the advantages of digital computing, current DNN-based AI methods require reshaping or even innovating this computing paradigm. AI has achieved dramatic progress in the last decade. AI is still in the phase of narrow AI, which demands large amount of manually labeled data to acquire knowledge of specialized tasks. In the next phase, we are expecting the broad AI that can adaptively and autonomously adapt to diversified tasks of various domains. Narrow AI is already computationally expensive in enormous scenarios. The vision of broad AI will even aggravate the computational predicament. Building efficient computing systems for such AI workload requires innovative reengineering of materials, architecture, and software.

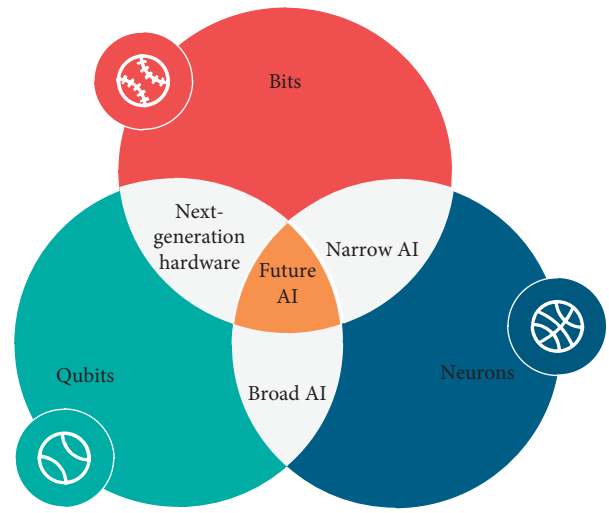

Figure 4: Road map to establish the next-generation AI-enabled computing systems.

The first category of solutions to AI-specific computing system stems from statistical and error-tolerant nature of deep learning. Such solutions sacrifice numerical precision for computational performance, yet generally achieve similar or even equivalent classification accuracy to the full-precision implementations [118–121]. We will witness a continuous decline in the precision demands of DNN training and inference in the coming decade. This trend is driven by the constant renovations of AI-specific digital hardware and matching algorithms, which will result in significant improvement in the performance of AI hardware.

As is previously discussed, another category of solutions lies in the idea of eliminating the overhead of data transmission between processing units and memory.

We can envision the high demands raise by DNN-based AI in the near future. Quantum computing enjoys the greatest computing power among almost all existing computing paradigms and thus has the potential to boost high-time-complexity deep learning applications that are knotty to the other computing paradigms.

4.3.3. Quantum Computing. Quantum computing generates an exponential state space of qubits (quantum bit) through exploring quantum superposition and entanglement. Computing power exponentially scales with the number of qubits: one additional qubit means doubled computing power. Prototypes of quantum computers have come out in the lab of hardware vendors like IBM [122, 123]. The next topic is to bridge the gap between the technical prototype

and real applications. For instance, quantum error correction (QEC) codes are indispensable for fault-tolerant quantum computing. Quantum computers will be a core accelerator of future AI-enabled computing systems. Nevertheless, currently, the cost of building a fault-tolerant quantum computing is beyond the reasonable range [124]. Further in-depth investigation is urgent.

4.3.4. Integration of Bits, Neurons, and Qubits. As aforementioned, a deep-learning-enabled computing system relies on three cornerstones: digital computing (bits), neural computing (neurons), and quantum computing (qubits). Systematic solutions to computational-resource-limited deep learning will require the integration of bits, neurons, and qubits. Bits can provide fundamental data storage and guarantee the robustness of underlying hardware. However, bits alone can only support programmed tasks for specific narrow purposes. Integrating neurons with bits generates narrow AI or even broad AI, which can not only distill insightful knowledge from unimaginably huge amount of data but also assist humans in a collaborative and more human-like manner. Various science and engineering problems are hopefully to be resolved with the assistance of AI. The core principle of a neural network is to search a function in the hypothesis space of the network and thus map a category of samples to a corresponding output label. Due to the large scale and complexity of science and engineering problems, a typical neural network necessarily requires a high capacity to generate a large hypothesis space. A large hypothesis space can possibly contribute to reducing the generalization error. Nevertheless, a large hypothesis space means more degree of freedom and demands a long time to let the stochastic-gradient-descent-impelled backpropagation find an approximation to the optimal solution. The exponentially scaling computing power just matches the time overhead of the similar order of magnitude.

Digital hardware like GPGPU and FPGA currently account for the mainstream accelerator of DNNs. Time-consuming manual fine-tuning of parallel code is an unavoidable operation to achieve optimal performance, with regard to every "DNN model-GPGPU type" pair. As a result, digital-hardware-accelerated DL is facing a barrier to efficient and agile programming. Moreover, the developing toolkit of analog-computing-enabled or quantum-computing-based deep learning is undoubtedly an essence when we someday handover analog computers or quantum computers to investigators, programmers, and computing resource providers.

5. Conclusion

In this paper, we investigate typical solutions of resource-limited deep learning and point out the open problems.

Existing solutions have achieved successes under specific scenarios. However, we expect future breakthroughs in the following two aspects. The first aspect is dedicated hardware. Most existing solutions depend on general-purpose digital hardware. Dedicated hardware, which takes into consideration unique characteristics of deep learning, is a promising direction to achieve further performance enhancements. The second aspect is the theoretical principles of deep learning. Simplifying the DNN is almost an inevitable method to reduce resource consumption. Nonetheless, such methods currently rely on empirical and iterative tuning. Additionally, the robustness of simplification is not theoretically guaranteed. Clarifying the theoretical principles of deep learning will enable more efficient simplification and guarantee robustness.

Disclosure

The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

Chunlei Chen conceptualized the study. Jiangyan Dai and Huihui Zhang were responsible for resources. Chunlei Chen and Peng Zhang prepared the original draft. Huixiang Zhang, Yugen Yi, and Yonghui Zhang reviewed and edited the manuscript.

Acknowledgments

This work was supported by the following research funds: the National Natural Science Foundation of China (31872847, 61471269, and 71661015), Industry-University Collaborative Education Program granted by Ministry of Education of China (201802217002), Planning Project of the 13th Five-year Plan of China Information Industry Association Education Branch (ZXXJ2019019), the Nature Science Foundation of Shandong Province (ZR2019PF023), the Higher Educational Science and Technology Program of Shandong Province (J18KA130 and J16LN56), the Science and Technology Development Program of Weifang (2019GX009, 2018GX004, and 2017GX002), the Key Technology Research and Development Program of Sichuan Province and Chengdu Municipality (szjj2015-054), the Doctoral Program of Weifang University (2016BS03 and 2015BS11), and the Science and Technology Benefiting People Plan Project of Weifang High Tech Zone (2019KJHM13).

References

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[2] H. W. Lin, M. Tegmark, and D. Rolnick, "Why does deep and cheap learning work so well?," *Journal of Statistical Physics*, vol. 168, no. 6, pp. 1223–1247, 2017.

[3] P. P. Brahma, D. Wu, and Y. She, "Why deep learning works: a manifold disentanglement perspective," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 10, pp. 1997–2008, 2016.

[4] Y. Bayle, M. Robine, and P. Hanna, "SATIN: a persistent musical database for music information retrieval and a supporting deep learning experiment on song instrumental classification," *Multimedia Tools and Applications*, vol. 78, no. 3, pp. 2703–2718, 2019.

[5] B. Xu, R. Cai, Z. Zhang et al., "NADAQ: natural Language database querying based on deep learning," *IEEE Access*, vol. 7, pp. 35012–35017, 2019.

[6] R. V. Swaminathan and A. Lerch, "Improving singing voice separation using attribute-aware deep network," in *Proceedings of the 2019 International Workshop On Multilayer Music Representation And Processing(MMRP)*, pp. 60–65, IEEE, Milano, Italy, 2019.

[7] M. S. Hossain, M. Al-Hammadi, and G. Muhammad, "Automatic fruit classification using deep learning for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1027–1034, 2019.

[8] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, "Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: a social multimedia perspective," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 566–578, 2019.

[9] Z. Huang, J. Tang, G. Shan, J. Ni, Y. Chen, and C. Wang, "An efficient passenger-hunting recommendation framework with multitask deep learning," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7713–7721, 2019.

[10] M. Zeng, M. Li, Z. Fei et al., "A deep learning framework for identifying essential proteins by integrating multiple types of biological information," *IEEE/ACM transactions on computational biology and bioinformatics*, p. 1, 2019.

[11] F. Pasa, V. Golkov, F. Pfeiffer, D. Cremers, and D. Pfeiffer, "Efficient deep network architectures for fast chest X-ray tuberculosis screening and visualization," *Scientific Reports*, vol. 9, no. 1, p. 6268, 2019.

[12] K. Li, J. Daniels, and C. Liu, "Convolutional recurrent neural networks for glucose prediction," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 2, pp. 603–613, 2019.

[13] A. Ramcharan, P. McCloskey, and K. Baranowski, "A mobile-based deep learning model for cassava disease diagnosis," *Frontiers in Plant Science*, vol. 10, p. 272, 2019.

[14] M. Reichstein, G. Camps-Valls, B. Stevens et al., "Deep learning and process understanding for data-driven earth system science," *Nature*, vol. 566, no. 7743, pp. 195–204, 2019.

[15] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, "Soft robot perception using embedded soft sensors and recurrent neural networks," *Science Robotics*, vol. 4, no. 6, Article ID eaav1488, 2019.

[16] W. Zheng, H. B. Wang, and Z. M. Zhang, "Multi-layer feed-forward neural network deep learning control with hybrid position and virtual-force algorithm for mobile robot obstacle avoidance," *International Journal of Control, Automation and Systems*, vol. 17, no. 4, pp. 1007–1018, 2019.

[17] Y. J. Heo, D. Kim, and W. Lee, "Collision detection for industrial collaborative robots: a deep learning approach," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 740–746, 2019.

[18] F. Niroui, K. Zhang, and Z. Kashino, "Deep reinforcement learning robot for search and rescue applications: exploration in unknown cluttered environments," *EEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 610–617, 2019.

[19] F. Ding, Z. Zhang, Y. Zhou, X. Chen, and B. Ran, "Large-scale full-coverage traffic speed estimation under extreme traffic conditions using a big data and deep learning approach: case study in China," *Journal of Transportation Engineering, Part A: Systems*, vol. 145, no. 5, Article ID 05019001, 2019.

[20] D. Mochizuki, Y. Abiko, T. Saito, D. Ikeda, and H. Mineno, "Delay-tolerance-based mobile data offloading using deep reinforcement learning," *Sensors*, vol. 19, no. 7, p. 1674, 2019.

[21] H. Ye, G. Y. Li, and B. H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.

[22] H. Ye and G. Y. Li, "Deep reinforcement learning based distributed resource allocation for V2V broadcasting," in *Proceedings of the 2018 14th International Wireless Communications And Mobile Computing Conference (IWCMC)*, pp. 440–445, Kansas City, MO, USA, 2018.

[23] W. Li, C. W. Pan, R. Zhang et al., "AADS.: Augmented autonomous driving simulation using data-driven algorithms," *Science Robotics*, vol. 4, no. 28, Article ID eaaw0863, 2019.

[24] S. M. Aldossari and K.-C. Chen, "Machine learning for wireless communication channel modeling: an overview," *Wireless Personal Communications*, vol. 106, no. 1, pp. 46–70, 2019.

[25] X. Qi, Y. Luo, G. Wu, K. Boriboonsomsin, and M. Barth, "Deep reinforcement learning enabled self-learning control for energy efficient driving," *Transportation Research Part C: Emerging Technologies*, vol. 99, pp. 67–81, 2019.

[26] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving [application notes]," *IEEE Computational Intelligence Magazine*, vol. 14, no. 2, pp. 83–98, 2019.

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

[28] A. Esteva, A. Robicquet, B. Ramsundar et al., "A guide to deep learning in healthcare," *Nature Medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[29] E. Kanjo, M. G. Y. Eman, and S. A. Chee, "Deep learning analysis of mobile physiological, environmental and location sensor data for emotion detection," *Information Fusion*, vol. 49, pp. 46–56, 2019.

[30] S. Chung, J. Lim, K. J. Noh, G. Kim, and H. Jeong, "Sensor data acquisition and multimodal sensor fusion for human activity recognition using deep learning," *Sensors*, vol. 19, no. 7, p. 1716, 2019.

[31] F. Mehmood, I. Ullah, S. Ahmad, and D. Kim, "Object detection mechanism based on deep learning algorithm using embedded IoT devices for smart home appliances control in CoT," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, 2019.

[32] M. Xu, F. Qian, M. Zhu, F. Huang, S. Pushp, and X. Liu, "DeepWear: adaptive local offloading for on-wearable deep learning," *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, pp. 314–330, 2020.

[33] A. Alelaiwi, "An efficient method of computation offloading in an edge cloud platform," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 58–64, 2019.

[34] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar, "An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices," in *Proceedings of the 2015 International Workshop on Internet of Things towards Applications—IoT-App '15*, pp. 7–12, ACM, Seoul, Korea, 2015.

[35] R. Affolter, S. Eggert, T. Sieberth, M. Thali, and L. C. Ebert, "Applying augmented reality during a forensic autopsy-Microsoft HoloLens as a DICOM viewer," *Journal of Forensic Radiology and Imaging*, vol. 16, pp. 5–8, 2019.

[36] C.-H. Wang, N.-H. Tsai, J.-M. Lu, and M.-J. J. Wang, "Usability evaluation of an instructional application based on Google Glass for mobile phone disassembly tasks," *Applied Ergonomics*, vol. 77, pp. 58–69, 2019.

[37] L. N. Huynh, Y. Lee, and R. K. Balan, "Deepmon: mobile GPU-based deep learning framework for continuous vision applications," in *Proceedings Of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 82–95, ACM, Niagara Falls, NY, USA, 2017.

[38] S. Lawrence, C. L. Giles, and A. C. Tsoi, "What size neural network gives optimal generalization? Convergence properties of backpropagation," Technical Report, NEC Research Institute, Princeton, NJ, USA, 1998.

[39] M. Dong, S. Wen, F. Zeng, Z. Yan, and T. Huang, "Sparse fully convolutional network for face labeling," *Neurocomputing*, vol. 331, no. 28, pp. 465–472, 2019.

[40] B. D. Rouhani, A. Mirhoseini, and F. Koushanfar, "Deep^3: leveraging three levels of parallelism for efficient deep learning," in *Proceedings of the 54th Annual Design Automation Conference 2017 on—DAC '17*, p.61, ACM, Austin, TX, USA, 2017.

[41] D. Stamoulis, E. Cai, D.-C. Juan, and D. Marculescu, "HyperPower: power-and memory-constrained hyper-parameter optimization for neural networks," in *Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 19–24, IEEE, Dresden, Germany, 2018.

[42] M. A. Hanif, M. U. Javed, R. Hafiz, S. Rehman, and M. Shafique, "Hardware-software approximations for deep neural networks," in *Approximate Circuits*, pp. 269–288, Springer, Berlin, Germany, 2019.

[43] A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-based accelerators of deep learning networks for learning and classification: a review," *IEEE Access*, vol. 7, pp. 7823–7859, 2018.

[44] D. Shin and H.-J. Yoo, "The heterogeneous deep neural network processor with a non-von Neumann architecture," *Proceedings of the IEEE*, pp. 1–16, 2019.

[45] F. Schuiki, M. Schaffner, F. K. Gurkaynak, and L. Benini, "A scalable near-memory architecture for training deep neural networks on large in-memory datasets," *IEEE Transactions on Computers*, vol. 68, no. 4, pp. 484–497, 2019.

[46] E. Azarkhish, D. Rossi, I. Loi, and L. Benini, "Neurostream: scalable and energy efficient deep learning with smart memory cubes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, pp. 420–434, 2018.

[47] H. Fuketa, H. Fuketa, T. Ikegami et al., "Image-classifier deep convolutional neural network training by 9-bit dedicated Hardware to realize validation accuracy and energy efficiency superior to the half precision floating point format," in *Proceedings Of the 2018 IEEE International Symposium On Circuits And Systems (ISCAS)*, pp. 1–5, IEEE, Florence, Italy, 2018.

[48] H. Tann, S. Hashemi, and S. Reda, "Lightweight deep neural network accelerators using approximate SW/HW techniques," in *Approximate Circuits*, pp. 289–305, Springer, Cham, Switzerland, 2019.

[49] A. G. Howard, M. Zhu, B. Chen et al., "Mobilenets: efficient convolutional neural networks for mobile vision applications," 2017, https://arxiv.org/abs/1704.04861.

[50] F. Chollet, "Xception: deep learning with depthwise separable convolutions," in *Proceedings Of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1251–1258, IEEE, Honolulu, HI, USA, 2017.

[51] A. Vasudevan, A. Anderson, and D. Gregg, "Parallel multi channel convolution using general matrix multiplication," in *Proceedings of the 2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 19–24, IEEE, Seattle, WA, USA, July 2017.

[52] M. Dong, S. Wen, Z. Zeng, Z. Yan, and T. Huang, "Sparse fully convolutional network for face labeling," *Neurocomputing*, vol. 331, no. 28, pp. 465–472, 2019.

[53] N. D. Lane, S. Bhattacharya, P. Georgiev et al., "Deepx: a software accelerator for low-power deep learning inference on mobile devices," in *Proceedings of the 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, p. 23, IEEE, Vienna, Austria, 2016.

[54] S. Ge, Z. Luo, Q. Ye, and X.-Y. Zhang, "MicroBrain: compressing deep neural networks for energy-efficient visual inference service," in *Proceedings of the 2017 IEEE Conference On Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1000-1001, IEEE, Atlanta, GA, USA, 2017.

[55] C. Deng, S. Liao, Y. Xie, K. K. Parhi, X. Qian, and B. Yuan, "PermDNN: efficient compressed DNN architecture with permuted diagonal matrices," in *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 189–202, IEEE, Fukuoka, Japan, 2018.

[56] W. Yang, L. Jin, S. Wang, Z. Cu, X. Chen, and L. Chen, "Thinning of convolutional neural network with mixed pruning," *IET Image Processing*, vol. 13, no. 5, pp. 779–784, 2019.

[57] R. Yazdani, M. Riera, J.-M. Arnau, and A. Gonzalez, "The dark side of DNN pruning," in *Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium On Computer Architecture (ISCA)*, pp. 790–801, IEEE, Los Angeles, CA, USA, 2018.

[58] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li, "Towards compact ConvNets via structure-sparsity regularized filter pruning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 2, pp. 574–588, 2019.

[59] M. Nazemi, G. Pasandi, and M. Pedram, "Energy-efficient, low-latency realization of neural networks through boolean logic minimization," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference on - ASPDAC '19*, pp. 274–279, ACM, Tokyo, Japan, 2019.

[60] B. D. Rouhan, A. Mirhoseini, and F. Koushanfar, "DeLight," in *Proceeding of the 2016 ACM/IEEE International Symposium On Low Power Electronics And Design*, pp. 112–117, ACM, San Francisco, CA, USA, 2016.

[61] J. Wang, A. Hertzmann, and D. J. Fleet, "Gaussian process dynamical models," *Advances in Neural Information Processing Systems*, vol. 19, pp. 1441–1448, 2006.

[62] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: a review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[63] P. P. Markopoulos, D. G. Chachlakis, and E. E. Papalexakis, "The exact solution to rank-1 L1-norm TUCKER2 decomposition," *IEEE Signal Processing Letters*, vol. 25, no. 4, pp. 511–515, 2018.

[64] Q. Cao, N. Balasubramanian, and A. Balasubramanian, "MobiRNN: efficient recurrent neural network execution on mobile GPU," in *Proceedings of the 1st International Workshop on Deep Learning for Mobile Systems and Applications—EMDL '17*, pp. 1–6, ACM, Niagara Falls, New York, USA, 2017.

[65] L. Jing, M. K. Ng, and T. Zeng, "Dictionary learning-based subspace structure identification in spectral clustering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 8, pp. 1188–1199, 2013.

[66] L. He and H. Zhang, "Iterative ensemble normalized cuts," *Pattern Recognition*, vol. 52, pp. 274–286, 2016.

[67] L. He, N. Ray, Y. Guan, and H. Zhang, "Fast large-scale spectral clustering via explicit feature mapping," *IEEE Transactions on Cybernetics*, vol. 49, no. 3, pp. 1058–1071, 2019.

[68] C. Wu, L. Zhang, Q. Li, Z. Fu, W. Zhu, and Y. Zhang, "Enabling flexible resource allocation in mobile deep learning systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 2, pp. 346–360, 2019.

[69] W. Haensch, T. Gokmen, and R. Puri, "The next generation of deep learning hardware: analog computing," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 108–122, 2019.

[70] D. Han, J. Lee, J. Lee, and H.-J. Yoo, "A low-power deep neural network online learning processor for real-time object tracking application," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 5, pp. 1794–1804, 2019.

[71] E. J. Fuller, S. T. Keene, A. Melianas et al., "Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing," *Science*, vol. 364, no. 6440, pp. 570–574, 2019.

[72] T. Tsuchiya, T. Tsuruoka, S. J. Kim et al., "Ionic decision-maker created as novel, solid-state devices," *Science Advances*, vol. 4, no. 9, Article ID eaau2057, 2018.

[73] C. Rios, N. Youngblood, Z. Cheng et al., "In-memory computing on a photonic platform," *Science Advances*, vol. 5, no. 2, Article ID eaau5759, 2019.

[74] J. Feldmann, N. Youngblood, C. D. Wright, H. Bhaskaran, and W. H. P. Pernice, "All-optical spiking neurosynaptic networks with self-learning capabilities," *Nature*, vol. 569, no. 7755, pp. 208–214, 2019.

[75] X. Gao, Z. Y. Zhang, and L. M. Duan, "A quantum machine learning algorithm based on generative models," *Science Advances*, vol. 4, no. 12, Article ID eaat9004, 2018.

[76] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[77] W. Zhizheng, *Automatic Speaker Verification Spoofing and Countermeasures Challenge (ASVspoof 2015) Database*, University of Edinburgh, The Centre for Speech Technology Research (CSTR), Edinburgh, UK, 2015.

[78] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, Granada, Spain, December 2011.

[79] A. Rakotomamonjy and G. Gasso, "Histogram of gradients of time-frequency representations for audio scene detection," 2014, https://arxiv.org/abs/1508.04909.

[80] Remote Sensing, 2015, http://www.ehu.es/ccwintco/index.php/HyperspectralRemoteSensingScenes.

[81] UCI machine learning repository, 2015, https://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities.

[82] UCI Machine Learning Repository, 2015, https://archive.ics.uci.edu/ml/datasets/isolet.

[83] J. Hays and A. Efros, "IM2GPS: estimating geographic information from a single image," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, June 2008.

[84] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *Proceedings of the First Workshop On Fine-Grained Visual Categorization, IEEE Conference On Computer Vision And Pattern Recognition*, Colorado Springs, CO, USA, June 2011.

[85] B. Thomee, B. Elizalde, D. A. Shamma et al., "Yfcc100M," *Communications of the ACM*, vol. 59, no. 2, pp. 64–73, 2016.

[86] J. Huang, V. Rathod, C. Sun et al., "Speed/accuracy trade-offs for modern convolutional object detectors," 2016, https://arxiv.org/abs/1611.10012.

[87] UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/UJIIndoorLoc.

[88] T. Weyand, I. Kostrikov, and J. Philbin, "PlaNet-photo geolocation with convolutional neural networks," *Computer Vision—ECCV 2016*, vol. 9912, pp. 37–55, Springer, Berlin, Germany, 2016.

[89] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits," 1998.

[90] F. Schroff, D. Kalenichenko, J. Philbin et al., "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, IEEE, Boston, MA, USA, June 2015.

[91] The CIFAR-10 Dataset, http://www.cs.toronto.edu/kriz/cifar.html.

[92] H. Bourlard and C. J. Wellekens, "Links between Markov models and multilayer perceptrons," *Advances in Neural Information Processing Systems*, vol. 1, pp. 502–510, 1988.

[93] Labeled Faces in the Wild, http://vis-www.cs.umass.edu/lfw/.

[94] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[95] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[96] K. He, X. Zhang, S. Ren et al., "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, IEEE, Las Vegas, NV, USA, 2016.

[97] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, IEEE, Boston, MA, USA, 2015.

[98] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," 2015, https://arxiv.org/abs/1506.02640.

[99] O. Russakovsky, J. Deng, H. Su et al., "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[100] PASCAL VOC2007, https://dbcollection.readthedocs.io/en/latest/datasets/pascal_voc2007.html.

[101] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: a dataset of 101 human actions classes from videos in the wild," 2012, https://arxiv.org/abs/1212.0402.

[102] S. Song, V. Chandrasekhar, N.-M. Cheung, S. Narayan, L. Li, and J.-H. Lim, "Activity recognition in egocentric life-logging videos," in *Proceedings of the Asian Conference On Computer Vision*, pp. 445–458, Springer, Singapore, 2014.

[103] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2014, https://arxiv.org/abs/1409.2329.

[104] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium, 2013.

[105] BVLC CaffeNet Model, https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet.

[106] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," 2015, https://arxiv.org/abs/1510.07945.

[107] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[108] M. J. Rasch, T. Gokmen, M. Rigotti et al., "RAPA-ConvNets: modified convolutional networks for accelerated training on architectures with analog arrays," *Frontiers in Neuroscience*, vol. 13, p. 753, 2019.

[109] A. Sebastian, M. Le Gallo, and E. Eleftheriou, "Computational phase-change memory: beyond von Neumann computing," *Journal of Physics D: Applied Physics*, vol. 52, no. 44, Article ID 443002, 2019.

[110] E. A. Cartier, W. Kim, N. Gong et al., "Reliability challenges with materials for analog computing?," in *Proceedings of the 2019 IEEE International Reliability Physics Symposium (IRPS)*, pp. 1–10, IEEE, Monterey, CA, USA, 2019.

[111] D. C. Mocanu, E. Mocanu, P. Stone et al., "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature Communications*, vol. 9, no. 1, p. 2383, 2018.

[112] S. Liu, D. C. Mocanu, A. R. R. Matavalam et al., "Sparse evolutionary deep learning with over one million artificial neurons on commodity hardware," 2019, https://arxiv.org/abs/1901.09181.

[113] T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: faster training without losing performance," 2019, https://arxiv.org/abs/1907.04840.

[114] R. Moradi, R. Berangi, and B. Minaei, "SparseMaps: convolutional networks with sparse feature maps for tiny image classification," *Expert Systems with Applications*, vol. 119, pp. 142–154, 2019.

[115] R. Ma, J. Miao, L. Niu, and P. Zhang, "Transformed ℓ1 regularization for learning sparse deep neural networks," *Neural Networks*, vol. 119, pp. 286–298, 2019.

[116] J. Yang and J. Ma, "Feed-forward neural network training using sparse representation," *Expert Systems with Applications*, vol. 116, pp. 255–264, 2019.

[117] D. Gil and W. M. J. Green, "The future of computing: bits + neurons + qubits," 2019, https://arxiv.org/abs/1911.08446.

[118] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalkrishnan, "Training deepneural networks with 8-bit floating point numbers," in *Proceedings of the 32nd Conference on Neural Information Processing Systems*, Montreal, Canada, 2018.

[119] C. Sakr and N. Wang, "Accumulation bit-width scaling for ultra-low precision training of deep networks," in *Proceedings of the International Conference On Learning Representations*, New Orleans, LA, USA, 2019.

[120] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalkrishnan, Z. Wang, and P. Chuang, "Accurate and efficient 2-bit quantized neural networks," in *Proceedings of the 2nd SysMLConference*, Stanford, CA, USA, 2019.

[121] K. Gopalkrishnan, "DNN training and inference with hyperscaled precision," in *Proceedings of the of Joint Workshop On-Device Machine Learning and Compact Deep Neural Network Representations*, Long Beach, CA, USA, 2019.

[122] IBM, *IBM Q Experience*, IBM, Armonk, NY, USA, https://www.ibm.com/quantumcomputing/technology/experience.

[123] IBM, *Quantum Computation Center Opens*, IBM, Armonk, NY, USA, 2019, https://www.ibm.com/quantum-computing/technology/experience.

[124] C. Vuillot, H. Asasi, Y. Wang et al., "Quantum error correction with the toric Gottesman-Kitaev-Preskill code," *Physical Review A*, vol. 99, no. 3, Article ID 032344, 2019.