RESEARCH ARTICLE OPEN ACCESS

Solving Real-Time Computational Problems using Smart Recognition AI System

Yew Kee Wong
JiangXi Normal University, Jiangxi - China

ABSTRACT
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.

Keywords — Blockchain, Artificial Intelligence, Autonomy, Cryptography, Privacy Protection

I. 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. Clarifies impetus of developing resource-limited DL. Representative solutions are discussed.

II. COMPUTATIONAL RESOURCE LIMITED CONTEXT OF DEEP LEARNING

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

III. INVESTIGATION ON EXISTING SOLUTIONS

B. *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 is the number of necessary training data points in the one-dimensional sample space, then the number of training data points is in *n*-dimensional sample space [38]. More training data points of higher dimension inevitably exacerbate overheads of memory, time, and energy.

C. *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.

D. *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 to reshape the DNN reference model under limited resource constraints [53]. By contrast to the clustering method of [39], 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, dramatically reduces the amount of DNN 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, 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 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 is fine-grained energy consumption control based on data dimension reduction. The framework 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 , where *x* is a data point in the design space of hyperparameters. Additionally, power and memory overhead is denoted as a function . Subsequently, hyperparameter tuning is formalized as an optimization problem: minimizing under the constraint that is lower than a threshold. Minimizing is costly due to the fact that has no close form. Consequently, adopts GP to approximate distributions of . Moreover, the framework leverages Bayesian optimization to iteratively select optimal hyperparameters and update distribution of . is assumed to obey Gaussian distribution. Let *y* denote the observations of . At the very beginning, an initial approximation of can be resolved as based on the assumption and a set of known 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 . And the selected *x* should push the 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 , and thus, is updated to .

E. 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 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. 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. divides each frame of image into equal-size blocks. 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, 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, 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, can adaptively adopt appropriate kernels at runtime so as to fit into a specific GPU with best efforts.

The main idea of 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 still adopts the idea of GPGPU-powered computing. However, exploit parallelism from three levels: data, network, and hardware. The ultimate goal of 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. 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. employs subroutines to perform hardware profiling. Each subroutine runs a specific operation with varying sizes on different platforms, separately. In this manner, 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. 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. 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 to capacitate commercial-off-the-shelf (COTS) mobile devices with the capability of adaptive resource scheduling [68]. Methods like try to compress the deep model. By contrast, 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, only needs to load currently required data from external storage into memory rather than hold entire data in memory throughout the execution period. Thus, remarkably lowers memory consumption. In addition, 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, is transparent to deep learning developers. It overloads default system functions of TensorFlow and Caffe. Developers can invoke 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.

F. 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 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.

IV. 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.

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," NEC Research Institute, Princeton, NJ, USA, 1998, Technical Report.

[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, "Deep3: leveraging three levels of parallelism for efficient deep learning," in *Proceedings of the 54th Annual Design Automation Conference 2017 on—DAC '17*, 61 pages, 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.

[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.