PERSPECTIVE
https://doi.org/10.1038/s41928-018-0059-3
nature electronics

Scaling for edge inference of deep neural networks

Xiaowei Xu[1], Yukun Ding[1], Sharon Xiaobo Hu[1], Michael Niemier[1], Jason Cong[2], Yu Hu[3] and Yiyu Shi[1]*

Deep neural networks offer considerable potential across a range of applications, from advanced manufacturing to autonomous cars. A clear trend in deep neural networks is the exponential growth of network size and the associated increases in computational complexity and memory consumption. However, the performance and energy efficiency of edge inference, in which the inference (the application of a trained network to new data) is performed locally on embedded platforms that have limited area and power budget, is bounded by technology scaling. Here we analyse recent data and show that there are increasing gaps between the computational complexity and energy efficiency required by data scientists and the hardware capacity made available by hardware architects. We then discuss various architecture and algorithm innovations that could help to bridge the gaps.

Deep neural networks (DNNs) have recently demonstrated performance comparable with, and in some cases superior to, that of human experts in areas such as image recognition[1–3] and the game of Go[4]. As the field develops, DNNs will be instrumental in delivering breakthroughs in various disciplines, including disease diagnosis, real-time language translation and autonomous driving[5–9]. Such accomplishments can be largely credited to ever-increasing computing power and a growing abundance of data. As larger clusters of faster computing nodes become available at lower cost and in smaller form factors, more data (assuming abundant data is readily available for the application of interest) can be used to train DNNs with more layers and neurons. This should translate to higher inference accuracy[10].

Network sizes have increased drastically over time, reaching beyond the petascale for some applications. Figure 1 shows, for example, how the number of parameters in popular deep neural networks has increased over the last twenty years, and clearly illustrates an exponential growth. For applications where powerful computing resources are easily accessible through network connections, large networks such as these may not present substantial challenges. However, for edge computation on embedded hardware platforms, where security, privacy and/or latency are critical considerations (such as smart sensors, wearable devices, autonomous driving and unmanned aerial vehicle tracking), inference must be performed locally or at the edge of the network, and such computation is subject to stringent area and power constraints due to the limited resources available.

To address the computational demands of ever-increasing network sizes, hardware architects have begun exploring techniques to compress DNNs for efficient edge inference. The ultimate judgment of such techniques is that lower power and area overheads can be achieved with minimal loss in inference accuracy. As many data scientists are focusing on increasing inference accuracy through designing more complex DNNs, there is a race between data scientists and hardware architects. Complementary metal–oxide semiconductor (CMOS) technology scaling based on Moore's law has provided hardware designers with a relatively easy path towards accommodating increasing network sizes. However, with the slowdown of CMOS scaling trends, novel architectures specifically designed for neural networks have emerged. In this Perspective, we demonstrate that gaps exist between Moore's-law-based CMOS scaling and the scaling of DNNs for edge inference, and discuss various architecture and algorithm innovations that could help to bridge these gaps.

The gaps

We examined recent data on state-of-the-art DNN accuracy and size, and the capacity of various hardware platforms. The results show that gaps exist between the pace of data scientists who design larger DNNs for better accuracy and that of hardware architects who try to accommodate them. In particular, Fig. 2 shows the top-five error rate (that is, the rate at which the corresponding class is not among the top five predictions) of the leading designs in the ImageNet classification competition[11] over recent years. This competition has made a substantial contribution to the sudden explosion of deep learning and has initiated many breakthroughs in DNN design[1,2,11–14]. As shown in Fig. 2, the top-five error rate has decreased exponentially over time, with a drop of approximately 30% each year. However, accompanying this trend is the drastically increased number of layers, parameters and number of operations, which can be seen from the theoretical upper and lower bounds of computation and memory complexity of DNNs with respect to desired accuracy[15–18]. The increased network complexity obviously calls for larger on-chip memory and higher I/O bandwidth for edge inference. We are particularly interested in whether hardware scaling can follow this trend in terms of performance and energy efficiency.

Performance gap. In Fig. 3a, we show the number of operations needed by the leading designs in the ImageNet classification competition against their top-five error rates. The number of operations increases exponentially from 1.4 gigaops per image (AlexNet, 2012) to 38 gigaops per image (VGG-19, 2014) as the top-five error rate drops from 16.4% to 7.32%. In 2014, GoogLeNet was developed, which uses parallel structural optimization[18,19] to concatenate multiple paths of different scales for more effective feature extraction. This innovation dramatically reduces the number of operations required with little drop in performance. However, the number of operations continued to increase exponentially as the top-five error rate further decreased to 3.08% (Inception-v4, 2016).

Graphics processing units (GPUs), field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) are popular hardware platforms for accommodating networks for edge inference[19–33]. Figure 3b depicts how the performance

[1]Department of Computer Science, University of Notre Dame, Notre Dame, IN, USA. [2]Department of Computer Science, University of California, Los Angeles, CA, USA. [3]School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China. *e-mail: yshi4@nd.edu

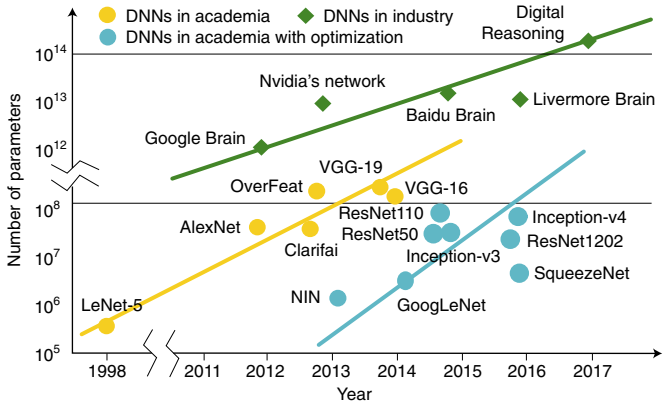

Fig. 1 | Scale of state-of-the-art DNNs. The parameter number of DNNs increases exponentially over time. The *y* axis is in log scale. Data taken from refs [116–120].

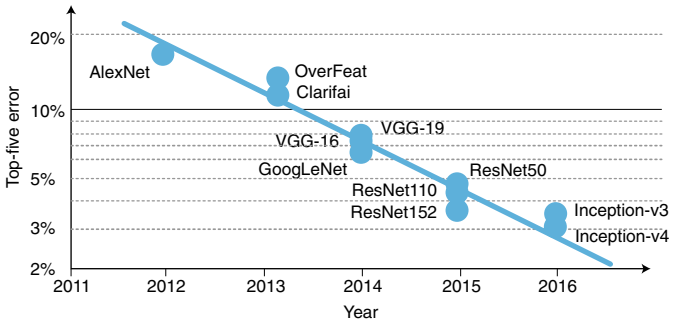

Fig. 2 | Top-five error rate of popular DNNs for ImageNet classification competition. The top-five error rate decreases exponentially over time. The *y* axis is in log scale. Data taken from refs [119,120].

density – the computation capacity per unit area, in gigaops per second per square millimetre – has evolved over time for the leading platforms in industry and academia. We note that the performance density of high-end GPUs remained flat from 2011 to 2014, as the architecture and technology remained the same. In 2015 technology migrated from 28 nm to 20 nm and, as a result, the perform ance density improved by about twofold, which is consistent with Moore's law. However, when technology scaled down to 12 nm in 2017, a minor drop in performance density appeared. This is due to the fact that more area is needed to accommodate larger memory and higher-bandwidth I/O[34]. However, it then essentially saturated; we see an exponential growth in performance density between 2011 and 2013, after which point it remained almost constant as no further technology scaling or important architectural innovation was deployed. ASIC designs for efficient DNN computation vary based on the primary metric of concern, such as performance, power consumption or speed. For instance, Myriad 2, Eyeriss and EIE all target low-power DNNs for embedded platforms. However, ultimately the peak performance density attainable by ASICs can easily surpass that of FPGAs and GPUs due to customization. That being said, all hardware platforms are generally bounded by Moore's law[35]; based on these trends, the performance density of these techniques will no longer increase when Moore's law ends at around 5 nm (ref. [36]).

By comparing Fig. 3a and Fig. 3b, it is clear that the performance density of leading hardware platforms cannot keep up with the number of operations required for better accuracy. Simply increasing the area to accommodate larger DNNs is not a sustainable option because of the associated power and cost on the edge.

Energy efficiency gap. Similarly, we can also observe that an increasing gap exists between the size of a network and the energy efficiency of the memory required by a hardware platform to accommodate it. Figure 4a shows how the number of network parameters increases with reducing top-five error rate, again from leading DNN designs in ImageNet classification competition. An exponential increase can be observed, both before and after the appearance of parallel structural optimization.

The energy efficiency of static random-access memory (SRAM) and dynamic random-access memory (DRAM) for leading CPUs, GPUs and ASICs[37,38] is presented in Fig. 4b. We focus on SRAM and DRAM only because SRAM access is two to three orders of magnitude more energy-efficient than DRAM access and ALU operations, whereas DRAM is much more cost efficient than SRAM and is therefore used more[27]. Recent architectures[23] suggest that the overall energy consumption is balanced between DRAM and SRAM, and together they dominate the energy consumption of a hardware platform.

From 2012 to 2015, the energy efficiency of DRAM (including double data rate (DDR), low-power DDR, graphics DDR, and 3D DRAM such as wide I/O mobile DRAM, high-bandwidth memory and hybrid memory cubes) increased due to Moore's-law-based CMOS scaling. After 2015, CMOS scaling no longer provided improvements in either energy efficiency or memory density. Because SRAM is realized with CMOS transistors, its energy efficiency is typically bounded by Moore's law[39]. The empirical evidence revealed by Fig. 4 therefore suggests that a memory's energy efficiency cannot keep up with the increasing size of a network. This results in a rising energy demand to process the same task, thus substantially exceeding the limited energy budget for edge inference.

Recently, some works[15–18] have theoretically demonstrated that the bounds of network size and computation increase exponentially with the accuracy, which partially supports our observation on the exponential growth trends above.

Bridge the gaps

CMOS scaling does not offer much help in meeting the increasingly demanding requirements for computation density and energy efficiency, so innovations in architecture, circuit and device are required instead. We focus here on innovations from hardware architects and data scientists that jointly will bridge the gap. Note that many interesting architectures have been recently proposed, such as Brainwave[40] by Microsoft, but they are not for edge inference and are thus not included in our discussion.

Architecture innovations. The systolic array architecture[41] is a specialized form of parallel computing in which tightly coupled processing elements are connected to a small number of nearest-neighbours in a mesh-like topology. This architecture therefore has a very low amount of global data transfer and can achieve high clock frequency. It was recently used to exploit the data movements that are characteristic of matrix multiplications and convolution computations, achieving promising performance in hardware designs for DNNs[28,32,42]. Google's TPU[28,43] and Cong and colleagues[32] explored this architecture from the perspectives of industry and academia, both showing a great performance improvement compared with existing implementations. Zhang et al.[42] further improved the energy efficiency by leveraging voltage-underscaling-based timing speculation. However, the systolic array architecture suffers from scalability issues because the shape of a systolic array is fixed in each particular implementation.

Near-data processing improves the energy efficiency of data movement by placing computing units near data. Integrating DRAM on the chip reduces the effort needed to move the data, thereby resulting in higher energy efficiency. Commercially available 3D DRAMs, hybrid memory cubes and high-bandwidth memory have been stacked on chip to reduce the capacitance of

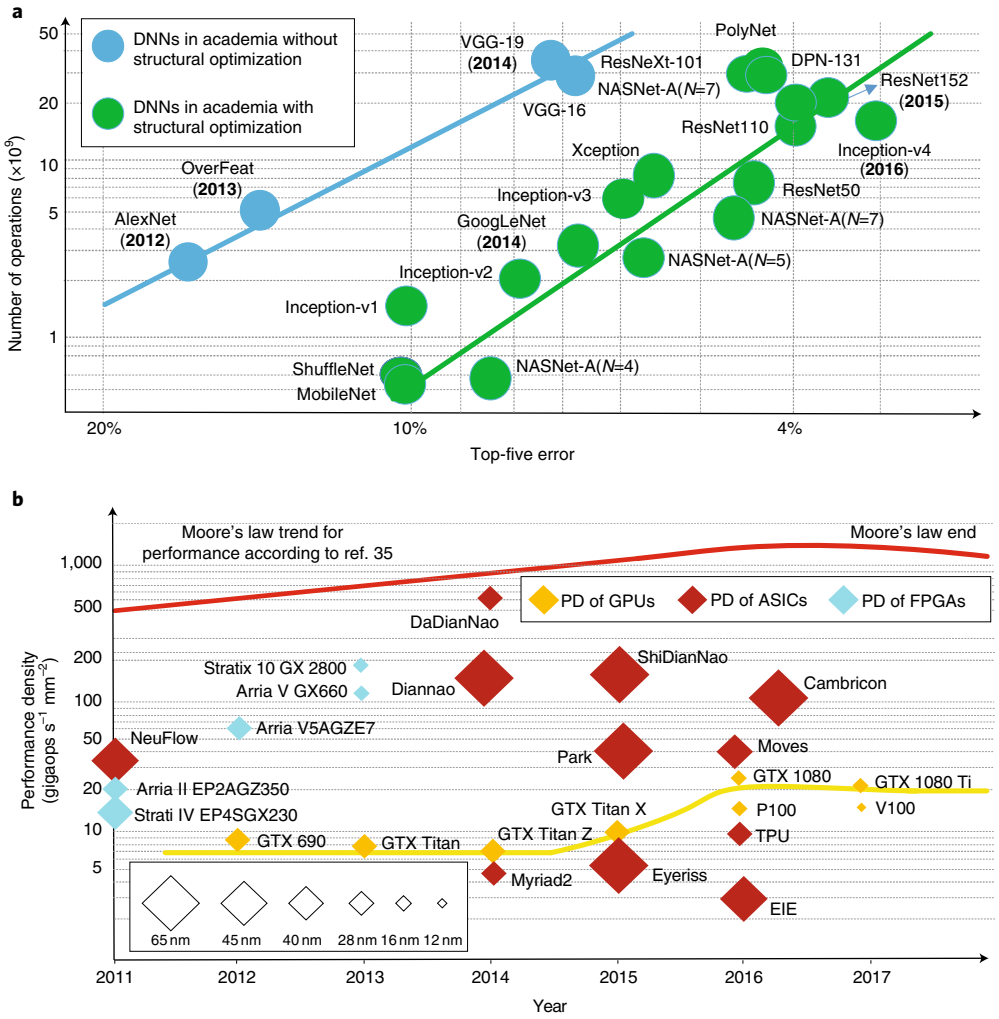

Fig. 3 | Gap between required number of operations and performance density. a, Number of operations versus top-five error rate for leading DNN designs from ImageNet classification competition. **b**, Performance density (PD) of leading GPU, ASIC and FPGA platforms. To catch up with the required number of operations, simply increasing the chip area is not feasible. Only the leading DNNs are labelled with a year. The *y* axis is in log scale. Data taken from refs [1–31,119,120].

the interconnect. Neurocube[44] stacked the hybrid memory cube die on a single-instruction multiple-data processor, while TETRIS[45] combined a hybrid memory cube with a spatial architecture. Unlike general DNN accelerators, near-data processing achieves optimal efficiency by using more area for computing. In order to achieve higher efficiency, some works have even moved the DRAM on chip. DaDianNao[23] adopted embedded DRAM for high-density on-chip memory, which achieves a 150-fold reduction in energy at the cost of larger chip size. There are also some works that moved computing units to sensors, thereby further reducing the cost of memory access. ShiDiannao[26] put vision processing in the sensor with no DRAM, yielding a 63-fold improvement in energy efficiency. RedEye[46] even omitted analogue-to-digital conversion and performed DNN computation in the analogue domain at the sensor.

Non-von Neumann architectures have also been explored to reduce computation and memory consumption. One such approach adopts non-volatile resistive memories as programmable resistive elements. Because computation is performed in the analogue domain, it can be extremely fast with ReRAM arrays[47]. The approach also brings high density and high energy efficiency as computation and memory are packed in the same chip area, thereby involving minimal data movement. ISAAC[48] adopted multicycle approach to perform high-precision calculations with limited memory using 25.1 million memristors. PRIME[49] employed a large memristor array for multi-level computation. Jain et al.[50] and Wang et al.[51] proposed the use of spin-transfer torque magnetic RAM for DNN computation.

Recently, representative array-level demonstrations have been reported. These include IBM's 500 × 661 phase change memory array for handwritten-digit recognition using the Modified National Institute of Standards and Technology (MNIST) database[52], Tsinghua's 128 × 8 analogue resistive RAM array for face recognition[53], UCSB's 12 × 12 crossbar array for pattern recognition[54], and UCSB's floating-gate array for MNIST image recognition[55]. Non-von Neumann architectures with memristors have several drawbacks: a large analogue-to-digital/digital-to-analogue conversion overhead, limited size of the memristor array, and energy and time overheads for memristor writing. It was recently shown that the analogue-to-digital conversion overhead can be eliminated by training the networks in the analogue domain[54], and memristor writing can also be mitigated[56]. Although non-von Neumann architectures with non-volatile resistive memories have considerable potential in both performance and energy efficiency, a number of requirements are yet to be met: special materials and device engineering to support the requirements of synaptic devices, increased array size, DNN mapping and EDA tools, and large-scale prototype

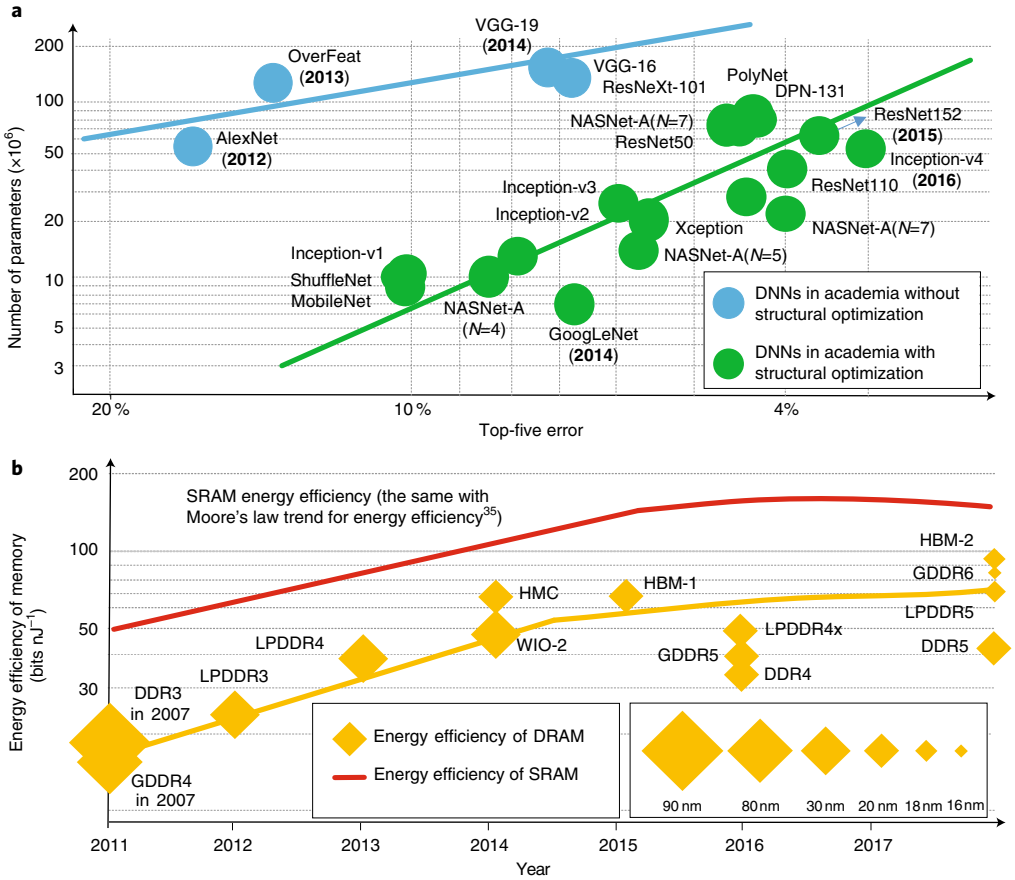

Fig. 4 | A gap exists between the required number of memory accesses and memory energy efficiency. a, Number of parameters (highly correlated with the number of memory accesses) versus top-five error rate and year for leading DNN designs from ImageNet classification competition. **b**, Memory energy efficiency of leading memory solutions. The memory efficiency cannot accommodate the increasing number of memory accesses with a limited energy budget. Only the leading DNNs are labelled with a year. The *y* axis is in log scale. Data taken from refs [1–31,34,37–39,119,120].

demonstrations. Some works have also performed multiply and accumulate operations in an SRAM array[57,58]. In particular, a bit-cell current can be used for computation, providing a 12-fold improvement in energy efficiency over von Neumann architectures.

Spiking neural networks, in contrast, model the behaviour of biological receptive fields and the human visual system, and have great potential for improving latency and energy efficiency. They typically adopt binary 'spikes' and low-resolution weights for high efficiency, but suffer from a lack of effective training algorithms owing to the non-differentiable nature of asynchronous spike events. Recently, several works have shed light on this challenge, including research on learning methods for achieving high accuracy in specific applications[59–62]. Hesham et al.[63] proposed an efficient temporal coding method for improving energy efficiency and reducing response time. Several hardware implementations have also been proposed, including TrueNorth by IBM[64,65], which verified the high energy efficiency and throughput of spiking neural networks on hardware. However, effective and generalized learning methods for large-scale spiking neural networks are still at an early stage of development.

Algorithm innovations. Because DNNs are both memory- and computation-intensive (convolutions, for example), many works have tried to reduce the amount of memory access or number of convolutional operations.

Sparsity optimization involves connection pruning and exploiting activation sparsity[66–68]. DNNs are frequently over-parameterized: that is, a large portion of the parameters are redundant and can be pruned. Recent works show that pruning can reduce 89% of memory accesses and 67% of computation operations[69,70]. Exploiting activation sparsity benefits from the fact that the rectified linear unit nonlinearity produces a large number of zero outputs, which can cut memory accesses and computation operations by half[24,71]. Ulrich et al.[72] explored sparsity with parameter sharing. A Bayesian point of view has also recently been explored for pruning and compression[73]. Hardware with customized data movement and control is required to support these reductions[27,74]. Jaderberg et al.[75] and Denton et al.[76] both used the low-rank approximation (LRA) to obtain a sparse convolution 2–4.5 times faster than without sparsity optimization with an accuracy loss of only 1%. However, the LRA suffers from a large number of hyper-parameters, which are difficult to train. To solve this problem, Wen et al.[77] proposed to obtain a compressed structure of deep CNNs by group Lasso regularization during the training, requiring only one hyper-parameter and providing a 3.1–5.1-fold improvement in speed over that without sparsity optimization. Wang et al.[78], Huang et al.[79] and Luo et al.[80] exploited the redundancy in the feature maps derived from the large number of filters in a layer. DeepRebirth[81] merged the consecutive non-tensor and tensor layers vertically or horizontally to reduce their number of layers, while Marc et al.[82] took the target domain into account for more compact LRA. Shufflenet[83] proposed two operations: point-wise group convolution and channel shuffle to reduce computation cost greatly while maintaining accuracy. Sotoudeh et al.[84] combined rank factorization with a reshaping process that added nonlinearity to the approximation function based on the LRA, obtaining a 2–14-fold improvement in speed over that without sparsity optimization.

Bit-width optimization aims to reduce the bit-width of parameters from floating point to fixed point[85]. Unlike sparsity optimization, it reduces the precision in exchange for memory access and computation operations with higher efficiency. Nonlinear quantization and the dynamic fixed point method[86–88] achieved four times less memory access than that without bit-width optimization with a top-five error rate of only 0.4–0.6% for ImageNet classification. Ternary weight network[89] and binaryConnect[90] further reduced the bit-width of weights to 2 bits or even 1 bit, but at the expense of a large reduction in accuracy. Recently, trained ternary quantization[91] and binary weight networks[92] have reduced this accuracy loss to only 0.6–0.8%. Some works have also achieved better accuracy by using nonlinear quantization to represent the parameter distribution[87,93,94]. Some studies also aim to quantize the activations[88,92,95–97]. Quantized neural networks[88], binarized neural networks[95] and XNOR-net[92] achieved a large reduction in memory/computation cost by reducing the weights to only 1 bit and the activations to 1–2 bits, although at the cost of substantially lower accuracy. Hu et al.[98] adopted hashing and Leng et al.[99] squeezed the last bit out for training to further improve the accuracy of binary weight networks, while Ko et al.[100,101] determined the optimal pair of weight/neuron bit precision by exploring their impact on performance. Bit-width optimization cannot compress DNNs arbitrarily small because at least 1 bit is always required. Note that in order to exploit precision reduction, customized hardware is also needed[102–106], such as Google's tensor processing units, which use 8 bits for computation[104], and Nvidia's 8-bit integer instructions for inference[105].

Researchers have also explored the computation of DNNs in other domains to reduce complexity. Motivated by the fact that complex convolutions in the time domain can be calculated with simple multiplications in the frequency domain, over the past a few years a few works have used fast Fourier transform (FFT)-based fast multiplication for DNNs[107,108]. Lecun et al.[109] applied the FFT to a single filter in the convolutional layer, with a relatively large loss in accuracy. Cheng et al.[110] calculated FFTs with a single circulant matrix to fully connected layers, resulting in a limited gain in weight reduction and performance. Cong and Xiao modelled the convolution computation as a special type of matrix–matrix multiplication and applied the Strassen algorithm to reduce the operation count[111]. Recently, Ding et al.[112] adopted block-circulant matrices in CirCNN to support both convolutional and fully connected layers. This system can reduce computation complexity from $O(n^2)$ to $O(n\log n)$ and storage complexity from $O(n^2)$ to $O(n)$ with almost no loss in accuracy. Following the same idea, Lu et al.[113] adopted the Winograd transformation, which is more hardware-friendly than the FFT.

Outlook

The approaches discussed here will only delay the widening of the gaps between Moore's-law-based CMOS scaling and DNN scaling where edge inference is required. To permanently close these gaps, we need new approaches that are fundamentally different from what exists today. The human brain is more than five orders of magnitude more energy efficient than all current DNNs[114,115], it does not require much training data with supervision to achieve high accuracy, nor does it need separate neural network structures for different tasks. With biology as our inspiration, it is clear we still have much room for improvement, and a long way to go.

Received: 11 December 2017; Accepted: 21 March 2018;
Published online: 17 April 2018

References

1. Krizhevsky, A. et al. ImageNet classification with deep convolutional neural networks. In *Adv. Neural Inf. Proc. Sys.* 1097–1105 (2012).
2. Szegedy, C. et al. Going deeper with convolutions. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition* 1–9 (2015).
3. He, K. et al. Identity mappings in deep residual networks. In *Eur. Conf. Computer Vision* 630–645 (Springer, 2016).
4. Silver, D. et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
5. Zhang, L. et al. Carcinopred-el: Novel models for predicting the carcinogenicity of chemicals using molecular fingerprints and ensemble learning methods. *Sci. Rep.* **7**, 2118 (2017).
6. Ge, G. et al. Quantitative analysis of diffusion-weighted magnetic resonance images: Differentiation between prostate cancer and normal tissue based on a computer-aided diagnosis system. *Sci. China Life Sci.* **60**, 37–43 (2017).
7. Egger, M. & Schoder, D. Consumer-oriented tech mining: Integrating the consumer perspective into organizational technology intelligence-the case of autonomous driving. In *Proc. 50th Hawaii Int. Conf. System Sciences* 1122–1131 (2017).
8. Rosenberg, C. Improving photo search: A step across the semantic gap. *Google Research Blog* (12 June 2013); https://research.googleblog.com/2013/06/improving-photo-search-step-across.html
9. Ji, S. et al. 3D convolutional neural networks for human action recognition. *IEEE T. Pattern Anal* **35**, 221–231 (2013).
10. Balluru, V., Graham, K. & Hilliard, N. Systems and methods for coreference resolution using selective feature activation. US Patent 9,633,002 (2017).
11. Sermanet, P. et al. Overfeat: Integrated recognition, localization and detection using convolutional networks. Preprint at https://arxiv.org/abs/1312.6229 (2013).
12. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. Preprint at https://arxiv.org/abs/1409.1556 (2014).
13. He, K. et al. Deep residual learning for image recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition.* 770–778 (2016).
14. Szegedy, C. et al. Rethinking the inception architecture for computer vision. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition.* 2818–2826 (2016).
15. Boris, H. Universal function approximation by deep neural nets with bounded width and ReLU activations. Preprint at https://arxiv.org/abs/1708.02691 (2017).
16. Liang, S. & Srikant, R. Why deep neural networks for function approximation? Preprint at https://arxiv.org/abs/1610.04161 (2016).
17. Dmitry, Y. Error bounds for approximations with deep ReLU networks. *Neural Networks.* **94**, 103–114 (2017).
18. Ding, Y. et al. On the universal approximability of quantized ReLU neural networks. Preprint at https://arxiv.org/abs/1802.03646 (2018).
19. https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units (2017).
20. Farabet, C. et al. Neuflow: A runtime reconfigurable dataflow processor for vision. *2011 IEEE Conf. Computer Vision and Pattern Recognition Workshops* 109–116 (2011).
21. Moloney, D. et al. Myriad 2: Eye of the computational vision storm. *Hot Chips 26 Symp*. 1–18 (2014).
22. Chen, T. et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *Proc. 19th Int. Conf. Architectural Support for Programming Languages and Operating Systems.* 269–284 (2014).
23. Chen, Y. et al. DaDianNao: A machine-learning supercomputer. *2014 47th Ann. IEEE/ACM Int. Symp. Microarchitecture.* 609–622 (2014).
24. Chen, Y. H., Krishna, T., Emer, J. S. & Sze, V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J. Solid-St. Circ.* **52**, 127–138 (2017).
25. Park, S. et al. A 1.93TOPS/W scalable deep learning/inference processor with tetra-parallel MIMD architecture for big-data applications. *2015 IEEE Int. Solid-St. Circ. Conf.* 1–3 (2015).
26. Du, Z. et al. ShiDianNao: Shifting vision processing closer to the sensor. *ACM SIGARCH Computer Architecture News* **43**, 92–104 (2015).
27. Han, S. et al. EIE: Efficient inference engine on compressed deep neural network. *2016 ACM/IEEE 43rd Ann. Int. Symp. Computer Architecture* 243–254 (2016).
28. Jouppi, N. P. et al. In-datacenter performance analysis of a tensor processing unit. *2017 ACM/IEEE 44th Ann. Int. Symp. Computer Architecture* 1–12 (2017).
29. Moons, B. & Verhelst, M. A 0.3–26 TOPS/W precision-scalable processor for real-time large-scale ConvNets. *IEEE Symp. VLSI Circuits* 1–2 (2016).
30. Liu, S. et al. Cambricon: An instruction set architecture for neural networks. *2016 ACM/IEEE 43rd Ann. Int. Symp. Computer Architecture* 393–405 (2016).
31. Whatmough, P. N. et al. 14.3 A 28nm SoC with a 1.2 GHz 568nJ/prediction sparse deep-neural-network engine with 0.1 timing error rate tolerance for IoT applications. *2017 IEEE Int. Solid-St. Circ. Conf.* 242–243 (2017).
32. Wei, X. et al. Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs. *2017 54th ACM/EDAC/IEEE Design Automation Conf.* **29**, 1–6 (2017).

33. Zhang, C. et al. Optimizing FPGA-based accelerator design for deep convolutional neural networks. *23rd Int. Symp. Field-Programmable Gate Arrays* https://doi.org/10.1145/2684746.2689060 (2015).
34. *NVIDIA TESLA P100* (NVIDIA, 2017); http://www.nvidia.com/object/tesla-p100.html
35. Sutter, H. The free lunch is over: A fundamental turn toward concurrency in software. *Dr Dobb's J.* **30**, 202–210 (2005).
36. Toumey, C. Less is Moore. *Nat. Nanotech.* **11**, 2–3 (2016).
37. Mutlu, O. Memory scaling: A systems architecture perspective. *Proc. 5th Int. Memory Workshop* 21–25 (2013).
38. *Using Next-Generation Memory Technologies: DRAM and Beyond* HC28-T1 (HotChips, 2016); available at https://www.youtube.com/watch?v=61oZhHwBrh8
39. Dreslinski, R. G., Wieckowski, M., Blaauw, D., Sylvester, D. & Mudge, T. Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits. *Proc. IEEE.* **98**, 253–266 (2010).
40. Microsoft unveils Project Brainwave for real-time AI. *Microsoft* (22 August 2017); https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwave/ (2017).
41. Kung, H. T. Algorithms for VLSI processor arrays. In *Introduction to VLSI Systems* 271–292 (1979).
42. Zhang, J., Ghodsi, Z., Rangineni, K. & Garg, S. Enabling extreme energy efficiency via timing speculation for deep neural network accelerators. *NYU Center for Cyber Security* (2017); http://cyber.nyu.edu/enabling-extreme-energy-efficiency-via-timing-speculation-deep-neural-network-accelerators/
43. *Cloud TPUs* (2017); https://ai.google/tools/cloud-tpus/
44. Kim, D., Kung, J., Chai, S., Yalamanchili, S. & Mukhopadhyay, S. Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory. *2016 ACM/IEEE 43rd Ann. Int. Symp. Computer Architecture* 380–392 (2016).
45. Gao, M., Pu, J., Yang, X., Horowitz, M. & Kozyrakis, C. TETRIS: Scalable and efficient neural network acceleration with 3D memory. *Proc. 22nd Int. Conf. Architectural Support for Programming Languages and Operating Systems* 751–764 (2017).
46. LiKamWa, R., Hou, Y., Gao, J., Polansky, M. & Zhong, L. RedEye: Analog ConvNet image sensor architecture for continuous mobile vision. *2016 ACM/IEEE 43rd Ann. Int. Symp. Computer Architecture* 255–266 (2016).
47. Li, C. et al. Analogue signal and image processing with large memristor crossbars. *Nat. Electron.* **1**, 52 (2018).
48. Ali, S. et al. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *2016 ACM/IEEE 43rd Ann. Int. Symp. Computer Architecture* 14–26 (2016).
49. Ping, C. et al. Prime: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. *2016 ACM/IEEE 43rd Ann. Int. Symp. Computer Architecture* 27–39 (2016).
50. Jain, S., Ranjan, A., Roy, K. & Raghunathan, A. Computing in memory with spin-transfer torque magnetic RAM. Preprint at https://arxiv.org/abs/1703.02118 (2017).
51. Kang, W., Wang, H., Wang, Z., Zhang, Y. & Zhao, W. In-memory processing paradigm for bitwise logic operations in STT–MRAM. *IEEE T. Magn.* **53**, 1–4 (2017).
52. Burr, G. W. et al. Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses), using phase-change memory as the synaptic weight element. *IEEE T. Electron Dev.* **62**, 3498–3507 (2015).
53. Yao, P. et al. Face classification using electronic synapses. *Nat. Commun.* **8**, 15199 (2017).
54. Prezioso, M. et al. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **521**, 61–64 (2015).
55. Guo, X. et al. Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology. *2017 IEEE Int. Electron. Dev. Meet.* 6.5.1–6.5.4 (2017).
56. Yu, S. et al. Binary neural network with 16 Mb RRAM macro chip for classification and online training. *2016 IEEE Int. Electron. Dev. Meet.* 16.2.1–16.2.4 (2016).
57. Zhang, J., Wang, Z. & Verma, N. In-memory computation of a machine-learning classifier in a standard 6T SRAM array. *IEEE J. Solid-St. Circ.* **52**, 915–924 (2017).
58. Jaiswal, A., Chakraborty, I., Agrawal, A. & Roy, K. 8T SRAM cell as a multi-bit dot product engine for beyond von-Neumann computing. Preprint at https://arxiv.org/abs/1802.08601 (2018).
59. Lee, J. H., Delbruck, T. & Pfeiffer, M. Training deep spiking neural networks using backpropagation. *Front. Neurosci.* **10**, 508 (2016).
60. O'Connor, P. & Max, W. Deep spiking networks. Preprint at https://arxiv.org/abs/1602.08323 (2016).
61. Hesham, M. Supervised learning based on temporal coding in spiking neural networks. *IEEE T. Neural Networks and Learning Systems* **PP**, 1–9 (2017).
62. Wen, W. et al. A new learning method for inference accuracy, core occupation, and performance co-optimization on TrueNorth chip. *2016 53nd ACM/EDAC/IEEE Design Automation Conf.* 1–6 (2016).
63. Mostafa, H., Pedroni, B. U., Sheik, S. & Cauwenberghs, G. Fast classification using sparsely active spiking networks. *2017 IEEE Int. Symp. Circuits and Systems* 1–4 (2017).
64. Qiao, N. et al. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Front. Neurosci.* **9**, 141 (2015).
65. Esser, S. K. et al. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proc. Natl Acad. Sci. USA.* **113**, 11441–11446 (2016).
66. Yu, R. et al. NISP: Pruning networks using neuron importance score propagation. Preprint at https://arxiv.org/abs/1711.05908 (2017).
67. Xu, X. et al. Empowering mobile telemedicine with compressed cellular neural networks. *IEEE/ACM Int. Conf. Computer-Aided Design* 880–887 (2017).
68. Xu, X. et al. Quantization of fully convolutional networks for accurate biomedical image segmentation. Preprint at https://arxiv.org/abs/1803.04907 (2018).
69. Han, S., Pool, J., Tran, J. & Dally, W. Learning both weights and connections for efficient neural network. *Proc. 28th Int. Conf. Neural Information Processing Systems* 1135–1143 (2015).
70. Yang, T. J., Chen, Y. H. & Sze, V. Designing energy-efficient convolutional neural networks using energy-aware pruning. Preprint at https://arxiv.org/abs/1611.05128 (2017).
71. Jorge, A. et al. Cnvlutin: Ineffectual-neuron-free deep neural network computing. *2016 ACM/IEEE 43rd Ann. Int. Symp. Computer Architecture* 1–13 (2016).
72. Ullrich, K., Meeds, E. & Welling, M. Soft weight-sharing for neural network compression. Preprint at https://arxiv.org/abs/1702.04008 (2017).
73. Louizos, C., Ullrich, K. & Welling, M. Bayesian compression for deep learning. *Proc. 30th Int. Conf. Neural Information Processing Systems* 3290–3300 (2017).
74. Brandon, R. et al. Minerva: Enabling low-power, highly-accurate deep neural network accelerators. *2016 ACM/IEEE 43rd Ann. Int. Symp. Computer Architecture* 267–278 (2016).
75. Jaderberg, M., Vedaldi, A. & Zisserman, A. Speeding up convolutional neural networks with low rank expansions. Preprint at https://arxiv.org/abs/1405.3866 (2014).
76. Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y. & Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. *Proc. 27th Int. Conf. Neural Information Processing Systems* 1269–1277 (2014).
77. Wen, W., Wu, C., Wang, Y., Chen, Y. & Li, H. Learning structured sparsity in deep neural networks. *Proc. 29th Int. Conf. Neural Information Processing Systems* 2074–2082 (2016).
78. Wang, Y., Xu, C., Xu, C. & Tao, D. Beyond filters: Compact feature map for portable deep model. *Int. Conf. Machine Learning* 3703–3711 (2017).
79. Huang, Q., Zhou, K., You, S. & Neumann, U. Learning to prune filters in convolutional neural networks. Preprint at https://arxiv.org/abs/1801.07365 (2018).
80. Luo, J. H., Wu, J. & Lin, W. Thinet: A filter level pruning method for deep neural network compression. Preprint at https://arxiv.org/abs/1707.06342 (2017).
81. Li, D., Wang, X. & Kong, D. DeepRebirth: Accelerating deep neural network execution on mobile devices. Preprint at https://arxiv.org/abs/1708.04728 (2017).
82. Masana, M., van de Weijer, J., Herranz, L., Bagdanov, A. D. & Alvarez, J. M. Domain-adaptive deep network compression. *Network* **16**, 30 (2017).
83. Zhang, X., Zhou, X., Lin, M. & Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. Preprint at https://arxiv.org/abs/1707.01083 (2017).
84. Sotoudeh, M. & Sara S. B. DeepThin: A self-compressing library for deep neural networks. Preprint at https://arxiv.org/abs/1802.06944 (2018).
85. Hashemi, S., Anthony, N., Tann, H., Bahar, R. I. & Reda, S. Understanding the impact of precision quantization on the accuracy and energy of neural networks. *2017 Design, Automation & Test in Europe* 1474–1479 (2017).
86. Jiantao, Q. et al. Going deeper with embedded FPGA platform for convolutional neural network. *Proc. 2016 ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays* 26–35 (2016).
87. Han, S., Mao, H. & Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. Preprint at https://arxiv.org/abs/1510.00149 (2016).
88. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. & Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. Preprint at https://arxiv.org/abs/1609.07061 (2016).
89. Li, F., Zhang, B. & Liu, B. Ternary weight networks. Preprint at https://arxiv.org/abs/1605.04711 (2016).
90. Courbariaux, M., Bengio, Y. & David, J. P. BinaryConnect: Training deep neural networks with binary weights during propagations. *Proc. 28th Int. Conf. Neural Information Processing Systems* 3123–3131 (2015).

91. Zhu, C., Han, S., Mao, H. & Dally, W. J. Trained ternary quantization. Preprint at https://arxiv.org/abs/1612.01064 (2016).
92. Rastegari, M., Ordonez, V., Redmon, J. & Farhadi, A. XNORNet: ImageNet classification using binary convolutional neural networks. *Eur. Conf. Computer Vision* 525–542 (2016).
93. Miyashita, D., Lee, E. H. & Murmann, B. Convolutional neural networks using logarithmic data representation. Preprint at https://arxiv.org/abs/1603.01025 (2016).
94. Zhou, A., Yao, A., Guo, Y., Xu, L. & Chen, Y. Incremental network quantization: Towards lossless CNNs with low-precision weights. Preprint at https://arxiv.org/abs/1702.03044 (2017).
95. Courbariaux, M. & Bengio, Y. BinaryNet: Training deep neural networks with weights and activations constrained to +1 or –1. Preprint at https://arxiv.org/abs/1602.02830 (2016).
96. Zhou, S. et al. DoReFaNet: Training low bitwidth convolutional neural networks with low bitwidth gradients. Preprint at https://arxiv.org/abs/1606.06160 (2016).
97. Cai, Z., He, X., Sun, J. & Vasconcelos, N. Deep learning with low precision by half-wave Gaussian quantization. *2017 IEEE Conference on Computer Vision and Pattern Recognition* 5918–5926 (2017).
98. Hu, Q., Wang, P. & Cheng, J. From hashing to CNNs: Training BinaryWeight networks via hashing. Preprint at https://arxiv.org/abs/1802.02733 (2018).
99. Leng, C., Li, H., Zhu, S. & Jin, R. Extremely low bit neural network: Squeeze the last bit out with ADMM. Preprint at https://arxiv.org/abs/1707.09870 (2017).
100. Ko, J. H., Fromm, J., Philipose, M., Tashev, I. & Zarar, S. Precision scaling of neural networks for efficient audio processing. Preprint at https://arxiv.org/abs/1712.01340 (2017).
101. Ko, J. H., Fromm, J., Philipose, M., Tashev, I. & Zarar, S. Adaptive weight compression for memory-efficient neural networks. *2017 Design, Automation & Test in Europe* 199–204 (2017).
102. Chakradhar, S., Sankaradas, M., Jakkula, V. & Cadambi, S. A dynamically configurable coprocessor for convolutional neural networks. *Proc. 37th Int. Symp. Computer Architecture* 247–257 (2010).
103. Gysel, P., Motamedi, M. & Ghiasi, S. Hardware-oriented approximation of convolutional neural networks. Preprint at https://arxiv.org/abs/1604.03168 (2016).
104. Higginbotham, S. Google takes unconventional route with homegrown machine learning chips. *The Next Platform* (19 May 2016).
105. Morgan, T. P. Nvidia pushes deep learning inference with new Pascal GPUs. *The Next Platform* (13 September 2016).
106. Judd, P., Albericio, J. & Moshovos, A. Stripes: Bit-serial deep neural network computing. *IEEE Computer Architecture Lett.* **16**, 80–83 (2016).
107. Zhang, C. & Prasanna, V. Frequency domain acceleration of convolutional neural networks on CPU-FPGA shared memory system. *Proc. 2017 ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays* 35–44 (2017).
108. Andrew, K. Reducing deep network complexity with Fourier transform methods. Preprint at https://arxiv.org/abs/1801.01451 (2017).
109. Mathieu, M., Henaff, M. & LeCun, Y. Fast training of convolutional networks through FFTs. Preprint at https://arxiv.org/abs/1312.5851 (2013).
110. Cheng, Y. et al. An exploration of parameter redundancy in deep networks with circulant projections. *Int. Conf. Computer Vision* 2857–2865 (2015).
111. Cong, J. & Xiao, B. Minimizing computation in convolutional neural networks. *Proc. 24th Int. Conf. Artificial Neural Networks* **8681**, 281–290 (2014).
112. Ding, C. et al. CirCNN: Accelerating and compressing deep neural networks using block-circulant weight matrices. *Proc. 50th Ann. IEEE/ACM Int. Symp. Microarchitecture* 395–408 (2017).
113. Lu, L., Liang, Y., Xiao, Q. & Yan, S. Evaluating fast algorithms for convolutional neural networks on FPGAs. *25th IEEE Int. Symp. Field-Programmable Custom Computing Machines* 101–108 (2017).
114. Fischetti, M. Computers versus brains. Scientific American (1 November 2011); https://www.scientificamerican.com/article/computers-vs-brains/.
115. Meier, K. The brain as computer: Bad at math, good at everything else. *IEEE Spectrum* (31 May 2017); https://spectrum.ieee.org/computing/hardware/the-brain-as-computer-bad-at-math-good-at-everything-else
116. Hachman, M. Nvidia's GPU neural network tops Google. *PC World* (18 June 2013); https://www.pcworld.com/article/2042339/nvidias-gpu-neural-network-tops-google.html
117. Digital reasoning trains world's largest neural network. *HPC Wire* (7 July 2015); https://www.hpcwire.com/off-the-wire/digital-reasoning-trains-worlds-largest-neural-network/
118. Wu, H. Y., Wang, F. & Pan, C. Who will win practical artificial intelligence? AI engineerings in China. Preprint at https://arxiv.org/abs/1702.02461 (2017).
119. Sze, V., Chen, Y. H., Yang, T. J. & Emer, J. S. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE.* **105**, 2295–2329 (2017).
120. Canziani, A., Paszke, A. & Culurciello, E. An analysis of deep neural network models for practical applications. Preprint at https://arxiv.org/abs/1605.07678 (2016).

Author contributions

X.X. contributed to data collection, analysis and writing. Y.D. contributed to data collection. S.H., M.N., J.C. and Y.H. contributed to discussion and writing. Y.S. contributed to project planning, development, discussion and writing.

Competing interests

The authors declare no competing interests.

Additional information

Reprints and permissions information is available at www.nature.com/reprints.

Correspondence and requests for materials should be addressed to Y.S.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.