Machine learning is becoming increasingly important in this era of big data. It enables us to extract meaningful information from the overwhelming amount of data being generated and collected every day. This information can be used to analyze and understand the data to identify trends (e.g., surveillance and portable/wearable electronics) or to take immediate action (e.g., robotics/drones, self-driving cars, and smart Internet of Things). In many applications, embedded processing near the sensor is preferred over the cloud due to privacy or latency concerns or limitations in the communication bandwidth. However, sensor devices often have stringent constraints on energy consumption and cost in addition to throughput and accuracy requirements. Circuit designers can play an important role in addressing these challenges by developing energy-efficient platforms to perform the necessary processing for machine learning. In this article, we will give a short overview of the key concepts in machine learning, discuss its challenges particularly in the embedded space, and...
highlight various opportunities where circuit designers can help to address these challenges.

**Introduction**

Machine learning is needed to extract meaningful, and ideally actionable, information from the enormous amount of data that is being generated and collected every day. Given the sheer volume of data, the high energy cost of communication, and the often limited communication bandwidth, there is an increasing need to perform the analysis locally near the sensor rather than sending the raw data to the cloud. Enabling machine learning at the edge also addresses important concerns related to privacy, latency, and security. Accordingly, embedded machine learning has shown to be beneficial for many applications such as those in the multimedia and medical fields.

For instance, computer vision is a form of machine learning that extracts information from images and videos, which are arguably the largest portion of big data as they account for over 70% of today’s Internet traffic [1]. In many applications (e.g., measuring wait times in stores and traffic patterns), it is desirable to extract the meaningful information from the video at the image sensor rather than in the cloud to reduce the communication cost. For other applications such as autonomous vehicles, drone navigation, and robotics, local processing is desired since the latency and security risk of relying on the cloud are too high. However, video involves high-dimensional data, which is computationally expensive to process; thus, low-cost hardware to analyze video is challenging yet critical to enabling these applications. While there is a wide range of computer vision tasks [2], in this article, we will focus on image classification as a driving example, where the task is to determine the class of the object in an image (Figure 1).

Another important application is speech recognition, which enables seamless interaction with electronic devices, such as smartphones. Speech recognition is the first step before many other tasks such as machine translation and natural language processing. Low-power hardware for speech recognition is explored in [3] and [4].

In the medical field, there is a clinical need to collect long-term data to help detect/diagnose various diseases or monitor treatment. For instance, the constant monitoring of electrocardiogram or electroencephalogram signals can identify cardiovascular diseases or detect the onset of a seizure for epilepsy patients, respectively. In many cases, these devices are either wearable or implantable, and thus the energy consumption must be kept to a minimum. The
Machine learning is needed to extract meaningful, and ideally actionable, information from the overwhelming amount of data that is being generated and collected every day.

use of embedded machine learning to extract meaningful physiological signals and process them locally is explored in [5] and [6].

**Machine Learning Basics**

Machine learning is a form of artificial intelligence that can perform a task without being specifically programmed. Instead, it learns from previous examples of the given task during a process called *training*. After learning, the task is performed on new data through a process called *inference*. Machine learning is particularly useful for applications where the data is difficult to model analytically.

A typical machine learning pipeline for inference can be broken down into two steps as shown in Figure 2: feature extraction and *classification*. Approaches such as deep neural networks (DNN) blur the distinction between these steps.

**Feature Extraction**

Feature extraction is used to transform the raw data into meaningful representations for the given task. Traditionally, feature extraction was designed through a handcrafted process by experts in a given field. For instance, it was observed that humans are sensitive to edges (i.e., gradients) in an image. As a result, many well-known computer vision algorithms use image gradient-based features such as histogram of oriented gradients (HOG) [7] and scale invariant feature transform [8]. The challenge in designing these features is to make them robust to variations in illumination and noise.

**Classification**

The output of feature extraction is represented by a vector \( \mathbf{x} \) in Figure 2, which is mapped to a score of confidence using a classifier. Depending on the application, the score is either compared to a threshold to determine if an object is *present* or compared to the other scores to determine the object *class*.

Techniques for classification include linear methods such as support vector machine (SVM) [9] and Softmax and nonlinear methods such as kernel-SVM [9] and Adaboost [10]. Many of these classifiers compute the score

---

**FIGURE 1:** An example of an image classification task. The machine learning platform takes in an image and outputs the confidence scores for a predefined set of classes.

**FIGURE 2:** The inference pipeline.
using a dot product of the features \((\mathbf{x})\) and a set of weights \((\mathbf{w})\) (i.e., \(\sum \mathbf{w} \cdot \mathbf{x}\)). As a result, machine learning hardware research tends to focus on reducing the cost of a multiply and accumulate (MAC) operation.

Training involves learning these weights from a dataset. Inference involves performing a given task using the trained weights. In most cases, training is done in the cloud, while inference can happen in the cloud or locally on a device near the sensor. In latter case, the trained weights are downloaded from the cloud and stored on the device. Thus, the device needs to be programmable in order to support a reasonable range of tasks.

**Deep Neural Networks**

Rather than using handcrafted features, the features can be directly learned from the data, similar to the weights in the classifier, such that the entire system is trained end to end. These learned features are used in a popular form of machine learning called called Deep Neural Networks (DNNs), also known as deep learning [11]. DNNs deliver higher accuracy than handcrafted features, sometimes even better than human-level accuracy, on a variety of tasks by mapping inputs to a high-dimensional representation; however, it comes at the cost of high-computational complexity, resulting in orders of magnitude higher energy consumption than handcrafted approaches [12].

There are many forms of DNNs (e.g., convolutional neural networks and recurrent neural networks). For computer vision applications, DNNs are often composed of multiple convolutional (CONV) layers [13] as shown in Figure 3; each layer involves the application of multiple high-dimensional filters to the incoming data. With each layer, a higher-level abstraction of the input data, called a feature map, is extracted that preserves essential yet unique information. Modern DNNs are able to achieve superior performance by employing a very deep hierarchy of layers on the order of tens to hundreds.

The output of the final CONV layer is typically processed by fully connected (FC) layers for classification. In FC layers, the filter and input feature map are the same size so that there is a unique weight for each input feature value. In between CONV and FC layers, additional functions can be added, such as pooling and normalization [14]. In addition, a nonlinear function, such as a rectified linear unit (ReLU) [15], is applied after each CONV and FC layer. Overall, convolutions account for over 90% of the run time and energy consumption in modern DNNs for computer vision.

Table 1 compares modern DNNs, with a popular neural net from the 1990s, LeNet-5 [16]. Today’s DNNs use more layers (i.e., deeper) and are
several orders of magnitude larger in terms of compute and storage. A more detailed discussion on DNNs can be found in [17].

**Impact of Difficulty of Task on Complexity**

The difficulty of the task must be considered when comparing different hardware platforms for machine learning as the size of the classifier or network (i.e., number of weights) and the number of MACs tend to be larger for more difficult tasks and thus require more energy. For instance, the task of classifying handwritten digits from the MNIST dataset [23] is much simpler than classifying an object into one of a 1,000 classes in the ImageNet data set [22] (Figure 4). Accordingly, LeNet-5, which is designed for digit classification, requires much less storage and compute than the larger DNNs in Table 1, which are designed for the 1,000-class image classification task. Thus, hardware platforms should only be compared when performing machine learning tasks of similar difficulty and accuracy, ideally, the same task with the same accuracy.

**Challenges**

The key metrics for embedded machine learning are accuracy, energy consumption, throughput, and cost. The challenge is to address all these requirements concurrently.

As previously discussed, the accuracy of the machine learning algorithm should be measured for a well-defined task on a sufficiently large dataset (e.g., ImageNet). Energy consumption is often dominated by data movement as memory access consumes significantly more energy than computation [24]. This is particularly challenging for machine learning as the high-dimensional representation and filters increase the amount of data generated, and the programmability needed to support different applications, tasks, and networks means that the weights also need to be read and stored. In this article, we will discuss various methods that reduce data movement to minimize energy consumption.

The throughput is dictated by the amount of computation, which also increases with the dimensionality of the data. In this article, we will discuss various transforms that can be applied to the data to reduce the number of required operations.

The cost is dictated by the amount of storage required on the chip. In this article, we will discuss various methods to reduce storage costs such that the area of the chip is reduced, while maintaining low off-chip memory bandwidth.

Currently, state-of-the-art DNNs consume orders of magnitude higher energy than other forms of embedded processing (e.g., video compression) [12]. We must exploit opportunities at multiple levels of hardware design to address all these challenges and close this energy gap.

**Opportunities in Architectures**

The MAC operations in both the feature extraction (CONV layers in a DNN) and classification (for both DNN and handcrafted features) can be easily parallelized. Two common highly parallel compute paradigms that can be used are shown in Figure 5.

**CPU and GPU Platforms**

Central processing units (CPUs) and graphics processing units (GPUs)
use temporal architectures such as SIMD or SIMT to perform the MACs in parallel. All the arithmetic logic units (ALUs) share the same control and memory (register file). On these platforms, all classifications are represented by a matrix multiplication. The CONV layer in a DNN can also be mapped to a matrix multiplication using the Toeplitz matrix. Software libraries that optimize for matrix multiplications can be used to accelerate processing on CPUs (e.g., OpenBLAS and Intel MKL) and GPUs (e.g., cuBLAS and cuDNN). The matrix multiplications can be further sped up by applying transforms such as fast Fourier transform [25], [26] and Winograd [27] to the data to reduce the number of multiplications.

**Specialized Hardware**

Specialized hardware provides an opportunity to optimize the data movement (i.e., data flow) to minimize accesses from the expensive levels of the memory hierarchy and maximize data reuse at the low-cost levels of the memory hierarchy. Figure 6 shows the memory hierarchy of the spatial architecture in Figure 5, where each ALU processing element (PE) has a local memory (register file) on the order of several kilobytes and a shared memory (global buffer) on the order of several hundred kilobytes. The global buffer communicates with the off-chip memory (e.g., DRAM). Data movement is allowed between the PEs using an on-chip network to reduce accesses to the global buffer and the off-chip memory.

The data flows of all three types of data (feature map, filter weights, and partial sums) affect energy consumption. Various data flows have been demonstrated in recent works [28]–[39], which differ in terms of the type of data that moves and the type of data that remains stationary in the register file of the PE [40]. The row stationary data flow, which considers the energy consumption of all three data types, reduces the energy consumption by 1.4× to 2.5× compared to the other data flows for the CONV layers [41].

**Opportunities in Joint Algorithm and Hardware Design**

The machine learning algorithms can be modified to make them more hardware friendly by reducing computation, data movement, and storage requirements, while maintaining accuracy.

**Reduce Precision**

GPUs and CPUs commonly use a 32-b floating point as the default representation. For inference, it is possible to use fixed point with reduced bit width for energy and area savings, and increased throughput, without affecting accuracy.

For instance, for object detection using handcrafted HOG features, only 11 bits are required per feature vector and only 5 bits per SVM weight [42]. For DNN inference, recent commercial hardware uses 8-b integer operations [43]. Custom hardware can be used to exploit the fact that the minimum bit widths vary per layer for energy savings [44] or increased throughput [45]. With more significant changes to the network, it is possible to reduce the bit width of DNNs to 1-b at the cost of reduced accuracy [46], [47].

**Sparsity**

Increasing sparsity in the data reduces storage and computation cost. For SVM classification, the weights can be projected onto a basis such that the resulting weights are sparse for a 2× reduction in number of multiplications [42]. For feature extraction, the input image can be made sparse by preprocessing for a 24% reduction in power consumption [48].

For DNNs, the number of MACs and weights can be reduced by removing weights through a process called pruning. This was first explored in [49] where weights with minimal impact on the output were removed. In [50], pruning is applied to modern DNNs by removing small weights. However, removing weights does not necessarily lead to lower energy. Accordingly, in [51], weights are removed based on an energy model [52] to directly minimize energy consumption.

Specialized hardware in [42] and [53]–[55] exploits sparse weights for

---

**FIGURE 6:** Memory hierarchy and data movement energy for a spatial architecture [41].
increased speed or reduced energy consumption. In Eyeriss [53], the PEs are designed to skip reads and MACs when the inputs are zero, resulting in a 45% energy reduction. In [42], specialized hardware is designed to avoid computation and storage of zero-valued weights, which reduces the energy and storage cost by 43% and 34%, respectively.

**Compression**

Lightweight compression can be applied to exploit data statistics (e.g., sparsity) to further reduce data movement and storage cost. Lossless compression can reduce the transfer of data on and off chip by around 2 times as shown in [5], [44], and [55]. Lossy compression such as vector quantization can also be used on feature vectors [42] and weights [3], [6], [56] such that they can be stored on chip at low cost. Note that when lossy compression is used, it is also important to evaluate the impact on accuracy.

**Opportunities in Mixed-Signal Circuits**

Mixed-signal circuit design can be used to address the data movement between the memory and PE and also the sensor and PE. However, circuit nonidealities should be factored into the algorithm design, for instance, by reducing precision as discussed in the “Opportunities in Joint Algorithm and Hardware Design” section. In addition, since the training often occurs in the digital domain, the analog-to-digital converter (ADC) and the digital-to-analog converter (DAC) overhead should also be accounted for when evaluating the system.

While spatial architectures bring the memory closer to the computation (i.e., into the PE), there have also been efforts to integrate the computation into the memory itself. For instance, in [57] the classification is embedded in the SRAM [Figure 7(a)], where the bit-cell current is effectively a product of the value of the 5-b feature vector (WLDAC) that drives the word line (WL), and the value of the binary weight stored in the bit cell. The currents from bit cells in the column are added together to discharge the bit line (BL) by $\Delta V_{BL}$. This approach gives 12x energy savings over reading the 1-b weights from the SRAM.

Recent work has also explored the use of mixed-signal circuits to reduce the computation cost of the MAC. It was shown in [59] that performing the MAC using switched capacitors can potentially be more energy efficient than digital circuits at low bit widths despite ADC and DAC overhead. In [60] and [61], the matrix multiplication (with bit widths less than or equal to 8 bits) is integrated into the ADC; this also moves the computation closer to the sensor and reduces the number of ADC conversions by 21x.

To further reduce the data movement from the sensor, [62] proposed performing the entire CONV layer in the analog domain at the sensor. Similarly, in [63], the entire HOG feature is computed in the analog domain to reduce the sensor bandwidth by 96.5%.

**Opportunities in Advanced Technologies**

Advanced technologies can also be used to reduce data movement by moving the processing and memory closer together. For instance, embedded DRAM (eDRAM) and hyper memory cube (HMC) are explored in [39] and [64], respectively, to reduce the energy access cost of the weights in DNNs. The multiplication can also be directly integrated into advanced nonvolatile memories [65] by using them as resistive elements [Figure 7(b)]. Specifically, the multiplications are performed where the conductance is the weight, the voltage is the input, and the current is the output; the addition is done by summing the current using Kirchhoff’s current law. Similar to the mixed-signal circuits, the precision is limited, and the ADC and DAC overhead must be considered in the overall cost. DNN processing using memristors is demonstrated in [58] and [66], where the bit width of the memristors is restricted to between 2 to 4 bits.

The computation can also be embedded into the sensors. For instance, an angle sensitive pixels sensor can be used to compute the gradient of...
the image input, which, along with compression, reduces the sensor bandwidth by 10x [67]. Such a sensor can also reduce the computation and energy consumption of the subsequent processing engine [48], [68].

Summary
Machine learning is an important area of research with many promising applications and opportunities for innovation at various levels of hardware design. The challenge is to balance the accuracy, energy, throughput, and cost requirements.

Since data movement dominates energy consumption, the primary focus of recent research has been to reduce the data movement while maintaining accuracy, throughput, and cost. This means selecting architectures with favorable memory hierarchies, developing data flows that increase data reuse at the low-cost levels of the memory hierarchy. Joint design of algorithms and hardware can be used to reduce bit-width precision, increase sparsity, and compression to further reduce the data movement requirements.

Mixed-signal circuit design and advanced technologies can be used to move the computation closer to the source by embedding computation near or within the sensor and in the memories.

Finally, designers should also consider the interactions between the different levels of design. For instance, reducing the bit width through hardware-friendly algorithm design enables reduced precision processing with mixed-signal circuits and nonvolatile memories. Reducing the cost of memory access with advanced technologies could also result in more energy-efficient data flows.

Acknowledgments
Portions of this article contains excerpts from the invited paper “Hardware for Machine Learning: Challenges and Opportunities” that appeared at the 2017 IEEE Custom Integrated Circuits Conference [69]. The work was done in collaboration with Yu-Hsin Chen, Joel Emer, Amr Suleiman, Tien-Ju Yang, and Zhengdong Zhang.

References
[1] Complete visual networking index (VNI) forecast,” Cisco, June 2016.
[2] R. Szegedy, Computer Vision: Algorithms and Applications. Springer Science & Business Media, 2010.
[3] M. Price, J. Glass, and A. P. Chandrakasan,” A 6 mW, 5,000-word real-time speech recognizer using WFSF models,” IEEE J. Solid-State Circuits, vol. 50, no. 1, pp. 102–112, 2015.
[4] R. Yazdani, A. Segura, J.-M. Arnau, and A. Gonzalez,” An ultra low-power hardware accelerator for automatic speech recognition,” in Proc. MICRO, 2016, pp. 1–12.
[5] T.-C. Chen, T.-H. Lee, Y.-H. Chen, T.-C. Ma, T.-D. Chiang, C.-J. Chou, C.-H. Yang, T.-H. Lin, and L.-C. Chen,” 1.4W/channel 16-channel EEG/ECG processor for smart brain sensor SoC,” in Proc. Symp. VLSI, 2010, pp. 21–22.
[6] K. H. Lee and N. Verma,” A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals,” IEEE J. Solid-State Circuits, vol. 48, no. 7, pp. 1625–1637, 2013.
[7] N. Dalal and B. Triggs,” Histograms of oriented gradients for human detection,” in Proc. Conf. Computer Vision and Pattern Recognition, 2005, pp. 886–893.
[8] D. G. Lowe,” Object recognition from local scale-invariant features,” in Proc. Int. Conf. Computer Vision, 1999, pp. 1150–1157.
[9] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[10] R. E. Schapire and Y. Freund,” Boosting: Foundations and Algorithms. Cambridge, MA: MIT Press, 2012.
[11] Y. LeCun, Y. Bengio, and G. Hinton," Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, May 2015.
[12] A. Suleiman, Y.-H. Chen, J. Emer, and V. Sze," Towards closing the energy gap between HOG and CNN features for embedded vision,” in Proc. IEEE Int. Symp. Circuits and Systems, 2017, pp. 444–447.
[13] Y. LeCun, K. Kavukcuoglu, and C. Farabet,” Convolutional networks and applications in vision,” in Proc. IEEE Int. Symp. Circuits and Systems, 2010.
[14] S. Ioffe and C. Szegedy,” Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in Proc. Int. Conf. Machine Learning, 2015, pp. 448–456.
[15] V. Nair and G. E. Hinton," Rectified linear units improve restricted boltzmann machines," in Proc. Int. Conf. Machine Learning, 2010, pp. 807–814.
[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner,” Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[17] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer," Efficient processing of deep neural networks: a tutorial and survey," arXiv Preprint, arXiv:1703.09039, 2017.
[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton,” ImageNet classification with deep convolutional neural networks,” in Proc. Neural Information Processing Systems Conf., 2012, pp. 1097–1105.
[19] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learning Representations, 2015.
[20] C. Szegedy, et al.” Going deeper with convolutions,” in Proc. Conf. Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[21] K. He, X. Zhang, S. Ren, and J. Sun,” Deep residual learning for image recognition,” in Proc. Conf. Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei,” ImageNet large scale visual recognition challenge,” Int. J. Computer Vision, vol. 115, no. 3, pp. 211–252, 2015.
[23] C. J. B. Yann LeCun and C. Cortes. The MNIST database of handwritten digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[24] M. Horowitz,” Computing’s energy problem (and what we can do about it),” in Proc. IEEE Int. Solid-State Circuits Conf., 2014, pp. 10–14.
[25] M. Mathieu, M. Henaff, and Y. LeCun,” Fast training of convolutional networks through FTFs,” in Proc. Int. Conf. Learning Representations, 2016.
[26] C. Dubout and F. Fleuret,” Exact acceleration of linear object detectors,” in Proc. European Conf. Computer Vision, 2012, pp. 301–311.
[27] A. Lavin and S. Gray,” Fast algorithms for convolutional neural networks,” in Proc. Conf. Computer Vision and Pattern Recognition, 2016, pp. 4013–4021.
[28] M. Sankaradavas, V. Jakkula, S. Cadambi, S. Chakradas, I. Durandovic, E. Cosatto, and H. P. Graf,” A massively parallel coprocessor for convolutional neural networks,” in Proc. ASAP, 2009, pp. 53–60.
[29] V. Sriram, D. Cox, K. H. Tsoi, and W. Luk,” Towards an embedded biologically-inspired machine vision processor,” in Proc. Field-Programmable Technology, 2010, pp. 273–278.
[30] S. Chakradas, M. Sankaradavas, V. Jakkula, and S. Cadambi,” A dynamically reconfigurable coprocessor for convolutional neural networks,” in Proc. IEEE Int. Symp. Computer Architecture, 2010, pp. 247–257.
[31] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello,” A 240 G-ops/s mobile coprocessor for deep neural networks,” in Proc. Conf. Computer Vision and Pattern Recognition Workshop, 2014, pp. 682–687.
B. Moons and M. Verhelst, “A 0.3–2.6 L. Cavigelli, D. Gschwend, C. Mayer, S. Park, K. Bong, D. Shin, J. Lee, S. Choi, M. Peemen, A. A. A. A. Setio, B. Mesman, and A. Suleiman, Z. Zhang, and V. Sze, “A Y.-H. Chen, J. Emer, and V. Sze, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” in *IEEE J. Solid State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

J. Albericio, P. Judd, T. Hetherington, T. Amores, N. E. Jerg, and A. Moshovos, “Cnvlutin: Ineffectual-neuron-free deep neural network computing,” in *Proc. IEEE Int. Symp. Computer Architecture*, 2016, pp. 243–254.

S. Han, H. Mao, and W. J. Dally, “Deep Compression: compressing deep neural network with pruning, trained quantization and huffman coding,” in *Proc. Int. Conf. Learning Representations*, 2016.

J. Zhang, Z. Wang, and N. Verma, “A machine-learning classifier implemented in a standard 6T SRAM array,” in *Proc. Symp. VLSI*, 2016, pp. 1–2.

A. Shafiee, A. Nag, N. Muralemonohar, R. Balasubramonian, M. Hu, R. S. Williams, and V. Srikumar, “ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” in *Proc. IEEE Int. Symp. Computer Architecture*, 2016, pp. 14–26.

B. Murmann, D. Bankman, E. Chai, D. Miyashita, and L. Yang, “Mixed-signal circuits for embedded machine-learning applications,” in *Proc. Asilomar Conf.*, 2015, pp. 1341–1345.

J. Zhang, Z. Wang, and N. Verma, “A matrix-multiplying ADC implementing a machine-learning classifier directly with data conversion,” in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2015, pp. 1–3.

E. H. Lee and S. S. Wong, “A 2.5 GHz 7.7 TOPS/W switched-capacitor matrix multiplier with co-designed local memory in 40nm,” in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2016, pp. 418–419.

R. Li, P. Cao, M. Polansky, and L. Zhong, “RedEye ConvNet image sensor architecture for continuous mobile vision,” in *Proc. IEEE Int. Symp. Computer Architecture*, 2016, pp. 255–266.

D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, “Neurocube: A programmable digital neuromorphic architecture with high-density, 3D memory,” in *Proc. IEEE Int. Symp. Computer Architecture*, 2016, pp. 380–392.

S. Yu and P.-Y. Cho, “Emerging memory technologies: Recent trends and prospects,” *IEEE Solid-State Circuits Mag.*, vol. 8, no. 2, pp. 43–56, 2016.

P. Chi, S. Li, Z. Qi, P. Gu, C. Xu, Z. Zhang, J. Zhao, V. Liu, Y. Wang, and Y. Xie, “PRIME: A novel, processing-in-memory architecture for neural network computation in ReRAM-based main memory,” in *Proc. IEEE Int. Symp. Computer Architecture*, 2016, pp. 27–39.

A. Wang, S. Sivaramakrishnan, and A. Molnar, “A 180nm CMOS image sensor with on-chip optoelectronic image compression,” in *Proc. IEEE Custom Integrated Circuits Conf.*, 2012, pp. 1–4.

H. Chen, S. Jayasuriya, J. Yang, S. Sivaramakrishnan, A. Veeraraghavan, and A. Molnar, “ASP Vision: Optically computing the first layer of convolutional neural networks using angle sensitive pixels,” in *Proc. Conf. Computer Vision and Pattern Recognition*, 2016, pp. 903–912.

V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, “Hardware for machine learning: Challenges and opportunities,” in *Proc. IEEE Custom Integrated Circuits Conf.*, 2017, pp. 1–8.

**About the Author**

Vivienne Sze (sze@mit.edu) is an associate professor in the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology (MIT). Her research interests include energy-aware signal processing algorithms and low-power circuit and system design for multimedia applications such as machine learning, computer vision, and video coding. Prior to joining MIT, she was with Texas Instruments, where she developed algorithms and hardware for the latest video coding standard H.265/HEVC. She received the B.A.Sc. degree from the University of Toronto in 2004 and the S.M. and Ph.D. degrees from MIT in 2006 and 2010, respectively. She is a recipient of several awards, including the Google Faculty Research Award, the AFOSR Young Investigator Award, the DARPA Young Faculty Award, and the Jin-Au Kong Outstanding Doctoral Thesis Prize. She currently serves on the Technical Program Committee of VLSI Symposium and MICRO.