nature electronics

ARTICLES

https://doi.org/10.1038/s41928-020-0397-9

Three-dimensional memristor circuits as complex neural networks

Peng Lin[1,3], Can Li[1], Zhongrui Wang[1], Yunning Li[1], Hao Jiang[1], Wenhao Song[1], Mingyi Rao[1], Ye Zhuo[1], Navnidhi K. Upadhyay[1], Mark Barnell[2], Qing Wu[2], J. Joshua Yang[1]✉ and Qiangfei Xia[1]✉

Constructing a computing circuit in three dimensions (3D) is a necessary step to enable the massive connections and efficient communications required in complex neural networks. 3D circuits based on conventional complementary metal–oxide–semiconductor transistors are, however, difficult to build because of challenges involved in growing or stacking multilayer single-crystalline silicon channels. Here we report a 3D circuit composed of eight layers of monolithically integrated memristive devices. The vertically aligned input and output electrodes in our 3D structure make it possible to directly map and implement complex neural networks. As a proof-of-concept demonstration, we programmed parallelly operated kernels into the 3D array, implemented a convolutional neural network and achieved software-comparable accuracy in recognizing handwritten digits from the Modified National Institute of Standard and Technology database. We also demonstrated the edge detection of moving objects in videos by applying groups of Prewitt filters in the 3D array to process pixels in parallel.

Deep neural networks (DNN) are computationally heavy models with a huge number of functional and complex connections[1,2]. Data flows in DNNs are translated into massive physical instructions that run in digital computers, which consumes tremendous computing resources and imposes a performance ceiling due to the need to shuttle data between the processing and memory units[3]. Emerging non-volatile devices[4,5], such as memristors[6–8], can directly implement synaptic weights and efficiently carry out resource-hungry matrix operations inside ensemble arrays[9–17]. Moreover, neural networks could be fully placed in memristor-based systems and executed on site, transforming the computing paradigms from high-demand, sequential digital operations into an activation-driven, parallel analogue-computing network. Energy consumption in such a system mainly occurs in a small population of highly activated computing pathways sparsely distributed in DNNs (Supplementary Fig. 1), which gains orders of magnitude better energy efficiency[18].

Existing memristor-based systems are, however, built on regular two-dimensional (2D) crossbar arrays. Their simplified connections cannot efficiently implement the full topology of often more complex structures in DNNs. Ultimately, it is the connections between neurons, which generate the emergent functional properties and states, that define the capability of a neural network[19]. Therefore, a fundamental shift in architecture design to emphasize the functional connectivity of device arrays is required to unleash the full potential of neuromorphic computing.

In this article, we show that extending the array design into 3D enables a large number of functional connections, allowing a complex memristor neural network to be built. Using the convolutional neural network (CNN) as an example, we built an eight-layer prototype 3D array with purposely designed connections that are able to perform parallel convolutional kernel operations in the CNN. The 3D design provides a high degree of functional complexity, which goes beyond the purpose of increasing packing density in conventional 3D architectures[20–26] and opens a critical avenue to using memristors for neuromorphic computing.

Design of 3D CNN

CNN is one of the most popular neural networks with wide applications in artificial intelligence, including computer vision, natural language processing and decision-making[1,27]. State-of-the-art CNN models have achieved excellent performances in image classification tasks enabled through a hierarchy of convolutional kernels (or filters). Each convolutional layer processes image data through 2D convolution operations that apply spatially replicated kernels over local input windows and carry out dot-product calculations (Fig. 1a). Processing 2D convolutions in parallel can drastically accelerate the processing speed and eliminate unwanted storage of partial convolution results, a fundamental requirement to implement a fully activation-driven hardware network. Although an individual kernel operation can be reorganized and implemented by a conventional 2D crossbar array (Fig. 1b), the placement of pixel-wise parallel convolutions in the same array is challenging because of the fully connected topology (Supplementary Note 1 and Supplementary Fig. 2). Even though tiles of small arrays may be used for parallel operations (Fig. 1c), this is not a scalable solution for a larger-sized image because the high-volume data movement between these large groups of kernel arrays would require an excessive area occupation for interconnections. More importantly, the highly complex connections in CNN would inevitably require complex 3D structured back-end-of-line interconnects between 2D arrays (that is, interconnects with many metal layers) and yet only serve a single device layer in conventional 2D circuits (Supplementary Note 1).

Alternatively, casting each array design into 3D space makes it possible to conveniently place replicated kernels inside a single

[1]Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, USA. [2]Air Force Research Laboratory Information Directorate, Rome, New York, USA. [3]Present address: Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA. ✉e-mail: jjyang@umass.edu; qxia@umass.edu

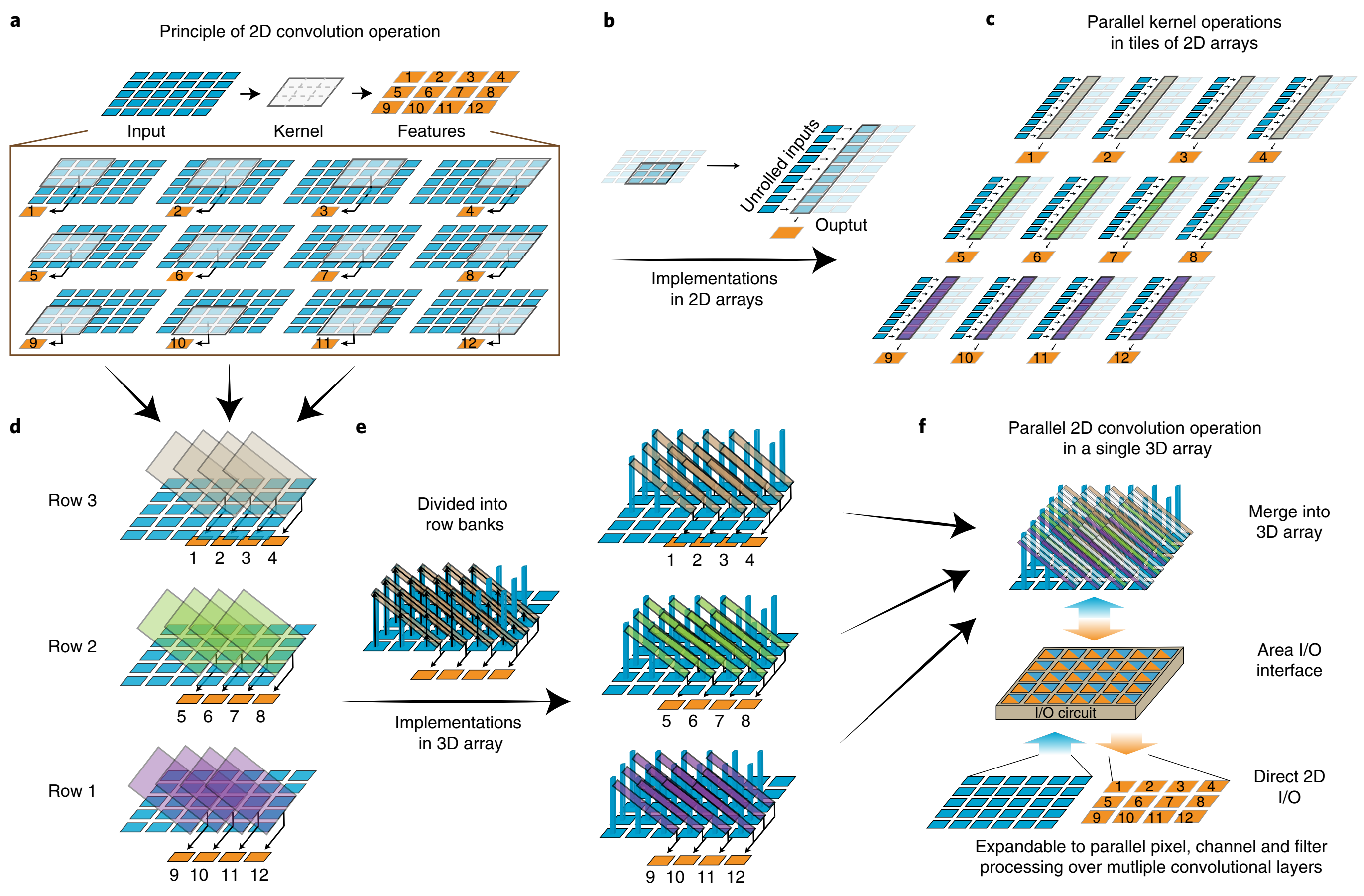
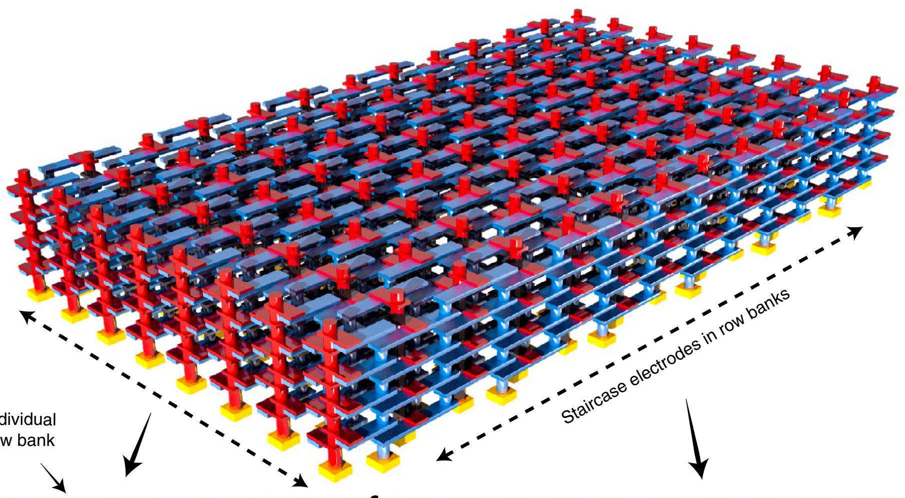

a
Principle of 2D convolution operation
Input
Kernel
Features
1 2 3 4 5 6 7 8 9 10 11 12
b
Unrolled inputs
Ouptut
Implementations in 2D arrays
c
Parallel kernel operations in tiles of 2D arrays
d
Row 3
Row 2
Row 1
e
Divided into row banks
Implementations in 3D array
f
Parallel 2D convolution operation in a single 3D array
Merge into 3D array
Area I/O interface
I/O circuit
Direct 2D I/O
Expandable to parallel pixel, channel and filter processing over mutliple convolutional layers

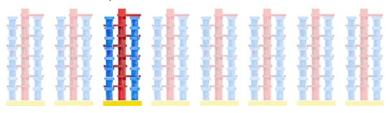
Fig. 1 | The purposely designed 3D array provides functional implementation of pixel-wise parallel convolutional kernel operations in a CNN. a, A 2D convolution operation in a convolutional layer applies spatially repeated kernel operations over local input pixels. **b**, Each kernel operation computes the dot product between the kernel weights and local input values. It is usually implemented in a 2D array by unrolling the data to $N \times 1$ vectors (N is pixel count). **c**, To process all the kernel operations in parallel, tiles of 2D arrays are needed, which becomes very area expensive for large inputs. **d**, 3D integration provides a design freedom to tightly place multiple kernels over the 2D input, and therefore enable a parallel processing capability of convolutions within the same array. **e**, Each kernel plane can be divided into individual row banks for a cost-effective fabrication and flexible operation. Input pillars are extended from the substrate to form memristor cells at the intersections to the 3D electrodes. **f**, Placement of all the row banks together implements complete 2D convolution operations in a single 3D array. 2D input and output are sent back and forth through the area I/O interface. The 3D array can be further expanded for parallel processing between different pixels, channels and filters over multiple convolutional layers.

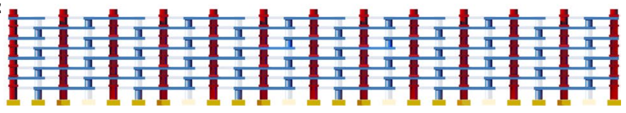
array and to offer a highly desirable area input/output (I/O) (horizontal planes of the 3D array) instead of a traditional perimeter I/O (edges of the 2D array), which provides highly compact and efficient implementations of CNNs. An example of such a 3D array for parallel convolution is shown in Fig. 1d–f. The projection of 12 localized kernels (marked in different colours for each row) over the input plane denotes the kernel operations (Fig. 1d), similar to the denotations in Fig. 1a. Benefitting from the design flexibility in 3D, parallelly operated kernels can partially overlap with each other without short-circuit issues, which leads to fine-grained parallel convolutions. This particular design serves the purpose of a proof-of-concept demonstration to prove the possibility of the functional design in a 3D memristor circuit (two alternative designs are shown in Supplementary Fig. 2). Each 3D kernel plane is divided into individual row banks (Fig. 1e) for a cost-effective fabrication and flexible operations (Supplementary Note 2). Inside each 3D row bank, memristors are formed at the cross-points between the pillar input electrodes and the 3D output electrodes, and their conductances are used as the weights of the high-density kernels in a convolutional layer. Merging all the row banks together forms the 3D array for a complete 2D convolution, as shown in Fig. 1f. In addition to pixel-wise parallel convolutions, our unique 3D design also offers several other highly rewarded advantages, which include (1) bilateral 2D data communications between the array and peripherals underneath, (2) handling most of the topological complexities in CNN so that the feature maps can be directly obtained at the output of the array with a minimal amount of postprocessing (for example, merging partial convolution results from local row banks) and (3) highly scalable and independent operations in row banks so that they can be flexibly programmed for different output pixels, filters or kernels from different convolutional layers (Supplementary Fig. 3). It becomes highly promising when row banks from different convolutional layers are placed tightly together and connected in the peripheral circuit underneath (assuming activations and pooling are done locally in the output circuit). Such a connection scheme offers substantial benefits in simplifying and shortening the massive and complex connections between convolutional layers.

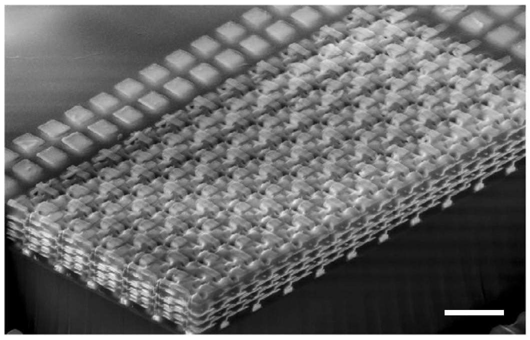
Eight-layer memristor array for computing

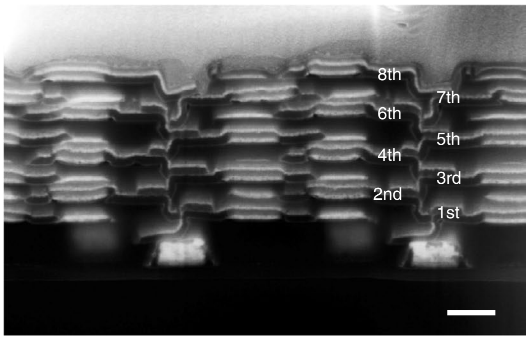
To experimentally demonstrate the feasibility of this 3D design, prototype 3D arrays were monolithically integrated on top of a substrate with I/O contact vias, as schematically illustrated in Fig. 2a–c. Different

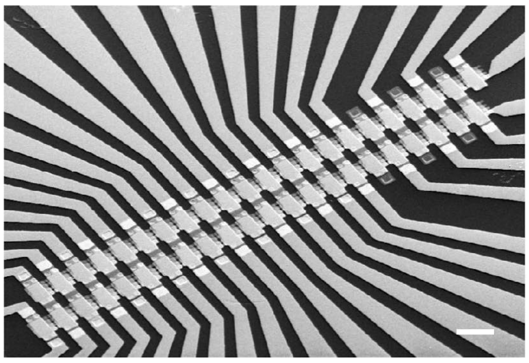
a


b


c


d


e


f


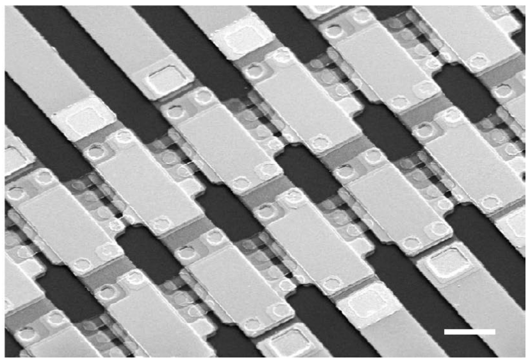
g


Fig. 2 | 3D monolithic integrated memristor circuits. a, Schematic of the 3D circuits composed of high-density staircase output electrodes (blue) and pillar input electrodes (red), which implement the 3D design in Fig. 1f. **b**, Sideview of 3D row banks. Each row bank in the 3D array operates independently. **c**, Sideview from column side showing unique staircase electrodes. Memristors with non-volatile current–voltage hysteresis are formed at the intersections between two electrodes and act as the synaptic weights. **d**, Scanning electron micrograph (SEM) of a representative 3D circuit with eight layers of monolithic integrated nanodevices with a feature size of 300 nm. Scale bar, 2 µm. **e**, Cross-sectional view of the circuits in **d**; this view from the lower-left angle shows the stacking of the eight layers. Scale bar, 300 nm. **f**, SEM of a 3D array with eight monolithic integrated layers of microscale devices. Scale bar, 40 µm. **g**, Magnified view of **f**. Scale bar, 15 µm.

3D row banks are physically isolated from each other (Fig. 2b), which may be electrically connected together via peripheral circuits for inference (Supplementary Fig. 3). Inside each row bank, the unique 3D topology is implemented by a non-orthogonal alignment between the input pillar electrodes (red) and output staircase electrodes (blue) that form dense but localized connections in the 3D array (Fig. 2c), which are not possible from standard 3D designs. The unique 3D layout, inspired by the 3D 'CMOL' architecture[28], possesses excellent stackability. Building each additional device layer only requires a short connection to the current top layer without the need for the complicated contact construction of conventional 3D designs[29]. Therefore, our design and process complexities do not scale-up with the number of 3D layers, which makes it a highly stackable approach. During stacking, only two sets of photomasks are required for all the layers

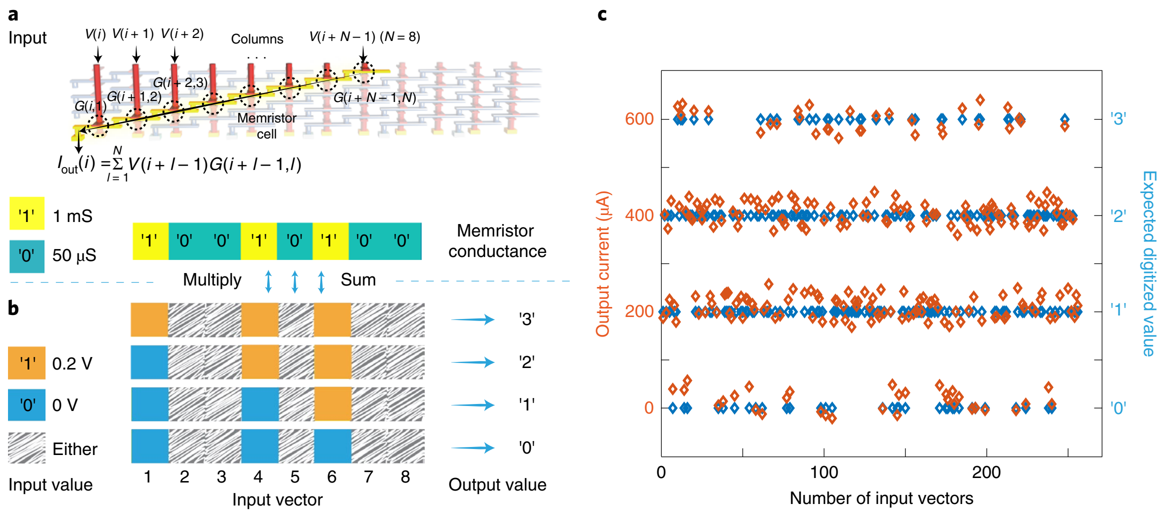


Fig. 3 | Principle of computing in our 3D array. a, Demonstration of the binary dot product process from the first output of the 3D array. The binary device weights read '10010100'. **b**, The digitized output value can be determined by the number of '1' devices that also received a '1' input. **c**, Experimental verification of the current sensed at the output by feeding input vectors from all the combinations of 8-bit binary values. The orange data points are the output currents and the blue data points are the desired normalized values. Large margins between the different states are clearly visible.

(one for the odd-numbered layers and the other for the even-numbered layers), which reduces the fabrication cost (Supplementary Note 2). As a demonstration, we fabricated 3D arrays that consisted of eight monolithically integrated device layers with feature sizes of 300 nm and 4 μm (Fig. 2d–g) patterned with electron beam lithography and photolithography, respectively (Methods and Supplementary Figs. 4 and 5). Each cell has a Pt/HfO_2/Ta memristor with a high reliability and excellent tunability[30]. The quasi-d.c. characteristics of test devices from all the layers showed a uniform and consistent switching behaviour (Supplementary Fig. 6).

It is worth mentioning that, despite the relative high wire resistance (60 Ω per block, 4-μm feature size) and linear current–voltage characteristic of the memristors, we still achieved successful array operations within this large 3D array (Fig. 2f). The sneak path problem is one of the most prominent challenges to scale-up the array size in conventional 2D or 3D arrays[29]. Usually, a selector device with a very high current–voltage non-linearity is required to connect in series with a memristor to suppress the sneak path current. However, so far there are no ideal selectors available with all the required properties. Even if such a selector existed, its large non-linearity would induce an unwanted non-linear amplification of noise and prevent Ohm's law being directly utilized for multiplication. In our 3D array, the number of memristors that share the same electrode is small regardless of the size of the entire 3D array. Consequently, the accumulated sneak path current during the writing and reading processes remains small even in a large 3D array, as confirmed by our HSPICE simulation (Supplementary Note 3 and Supplementary Figs. 7–9). This unique property enables excellent lateral scalability of our 3D array that cannot be achieved in any of existing selectorless 2D and 3D array designs.

Each convolution is a dot product between a matrix of weights (kernel) and a matrix of inputs (for example, pixels). It has been implemented in standard 2D memristor crossbar arrays, but each array can only process convolutions from the same input pixels due to the aforementioned full connectivity[31,32]. Our 3D array can provide a fine-grained, pixel-wise parallel-weighted sum operation, owing to the local connectivity of electrodes and high-density area-distributed I/Os. Using a single 3D row bank as an example, the output current at column i (Fig. 3a) is the dot product between input voltages (V_{in}) and memristor conductance (G) from column i to $i+N-1$:

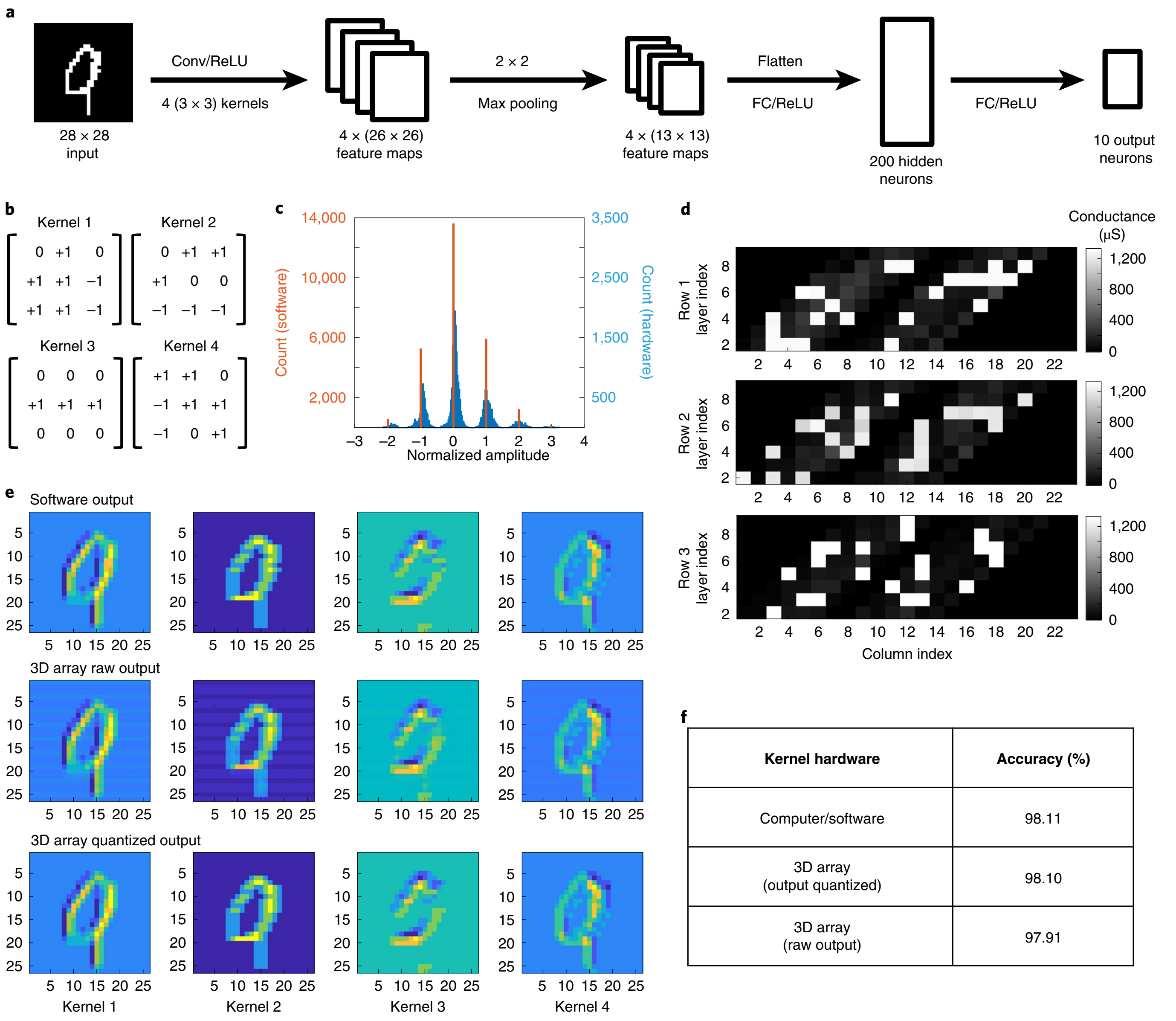
$$I_{out}(i) = \sum_{l=1}^{N} V_{in}(i+l-1)G(i+l-1,l) \qquad (1)$$

where l is the layer index and N is the total number of layers.

To demonstrate the computing principle, we used binary input signals and programmed the memristors to binary conductance states. Electrical accesses to the 3D array (programming and reading operations) were carried out using a customized measurement system (Methods and Supplementary Fig. 10). Taking the first output ($i=1$) in the 3D row bank as an example, the eight memristors, one per layer, were programmed to '10010100' in which '1' represents a high conductance (1 mS) and '0' a low conductance (50 μS) state (Fig. 3a). We fed an input vector that consisted of eight voltages (0.2 or 0 V) to the row bank (Fig. 3b). As there are only three memristors at high conductance states, the possible output current levels are 0, 200, 400 and 600 μA (or '0', '1', '2' and '3' after digitization), regardless of the number and order of the 0.2 V inputs applied (Fig. 3c). Although the output currents are still affected by non-ideal factors, such as variation in the device conductance, each individual level is clearly distinguished from others, which confirms that the array can be used for computing.

Parallel processing of images and videos

We built a four-layer CNN to classify binarized images from the Modified National Institute of Standard and Technology (MNIST) handwritten digits database. The CNN model consists of one convolutional layer with four kernels (3 × 3 with ternary weights), one 2 × 2 max pooling layer and one fully connected layer with 200 hidden neurons and one output layer with 10 output neurons (Fig. 4a and Supplementary Note 4). After offline training, the four convolutional kernels (Fig. 4b) were spatially replicated and programmed into the 3D array based on the algorithm discussed in Methods and Supplementary Fig. 11 as proof-of-concept demonstration of pixel-wise parallel operations. The rest of the network components, such as activation functions, pooling layer and fully connected layers, were implemented in software. The distribution of normalized output states from all the output electrodes and the programmed conductance map of the 3D circuit are shown in Fig. 4c,d,

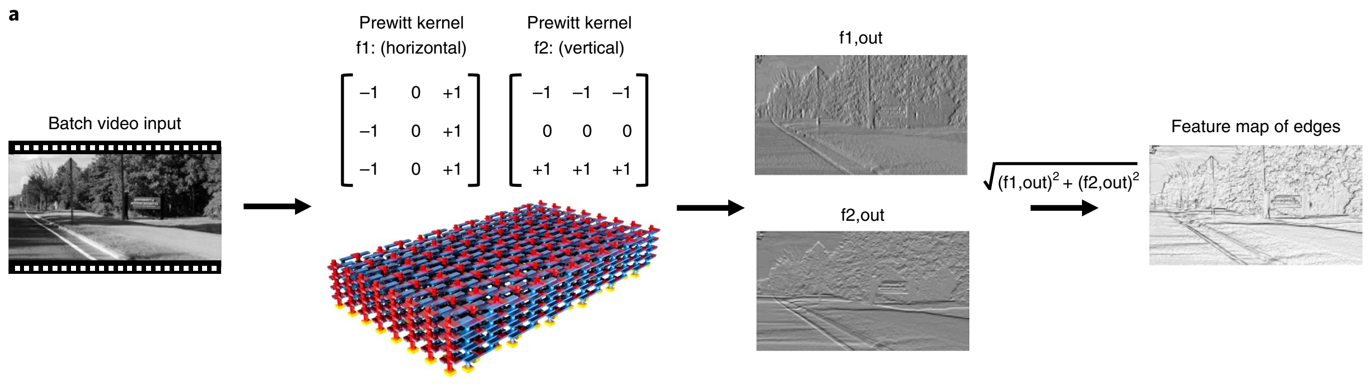


Kernel hardware	Accuracy (%)
Computer/software	98.11
3D array (output quantized)	98.10
3D array (raw output)	97.91

Fig. 4 | Parallel convolutional kernel operations using the 3D circuits. a, A four-layer CNN is used to classify handwritten digits in the MNIST database. **b**, Four 3×3 convolutional kernels were obtained by ex situ training in software and programmed into the 3D circuit to perform parallel processing of the input pixels. Ternary weights (−1,0,1) were used for forward and backward propagation, whereas double precision (64-bit floating point) was used for weight updating. Three duplicates of each kernel (12 in total) were parallel programmed into the 3D array for parallel processing. **c**, The distribution of all the array output (normalized) superimposed to the desired output states. The broadened distribution of output comes from non-ideal tuning of the weights, but the output still achieved an overall separation between different states. **d**, The measured conductance map of the three row banks with parallel-processed kernels. The algorithm used to assign the weights is described in Methods. **e**, Comparison between the software- and hardware-processed kernel outputs. The output features can be almost fully recovered after quantization. **f**, Inference accuracy on the MNIST test set. The quantized output from the 3D circuit achieved an almost identical performance as that of the software, and the raw output from the 3D circuit also shows a good robustness against noise. FC, fully connected; ReLU, rectified linear unit.

respectively. Different output states can still be distinguished in the appearance of the noises and variations. The output signal is nearly perfectly reproduced after quantization, as shown in Fig. 4e. The inference on the MNIST test set confirms an almost identical recognition accuracy between the software (98.11%) and hardware (98.10%) implemented kernels over the 10,000 test images (Fig. 4f). Meanwhile, even without quantization, we achieved a 97.91% classification accuracy, which shows its robustness against noise in the 3D array. In the future, the fully connected layers could also be implemented in memristor arrays to form a highly efficient hybrid 3D/2D system for activation-driven analogue computing (speed and energy estimation are shown in Supplementary Note 5 and Supplementary Tables 1 and 2, and a preliminary demonstration is shown in Supplementary Note 6 and Supplementary Fig. 12). It generally requires large fully connected arrays to achieve highly accurate classifications, and yet the majority of computations are still done in 3D-array based convolutional layers.

Finally, we employed the 3D circuits for parallel video processing. Two Prewitt filters, which are popular operators for image-edge detection, were programmed into the 3D array using the



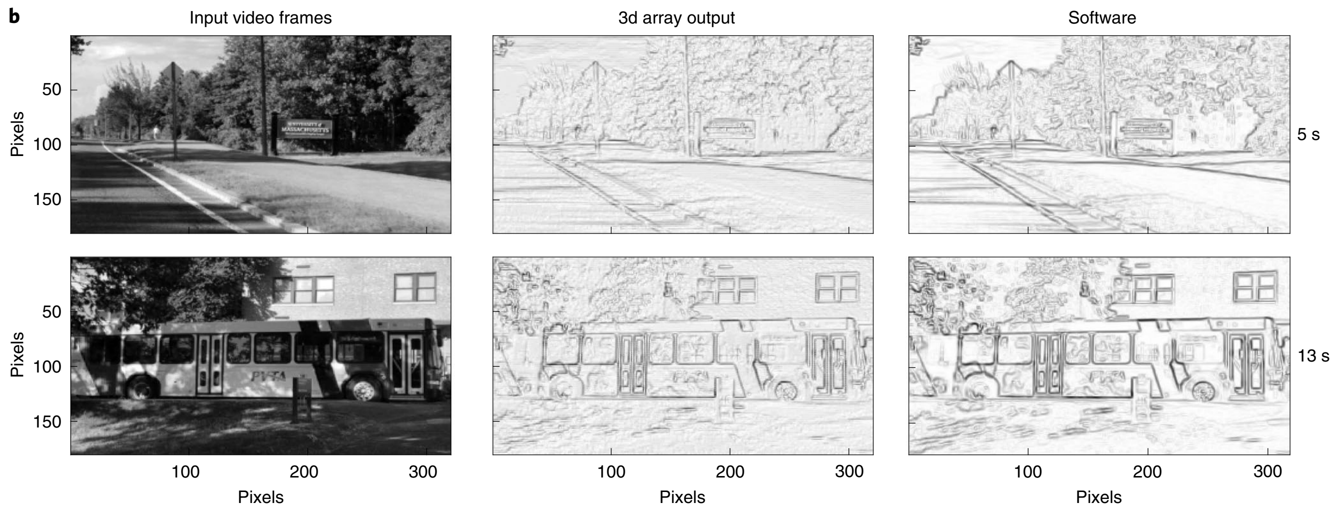


Fig. 5 | Parallel video processing using the 3D circuits. a, Two sets of Prewitt kernels (f1 and f2) are programmed into the 3D circuits to perform parallel edge detection. **b**, Comparison between the hardware and software edge detection of video frames. The hardware-processed video shows comparable results with fine edge features extracted from the input video (Supplementary Video 1).

same approach for the CNN (Methods) (Fig. 5a). The output of each kernel was recorded and combined in postprocessing to produce the final video frames. Fine edge features were successfully extracted using the 3D array and comparable results were again obtained between the software- and hardware-implemented kernel operations (Fig. 5b). One potential benefit of using our 3D array for video processing, when achieved with a large-scale integrated design, is that it hosts all the computation in real time, and can directly interface with a 2D image sensor array through kernel vertical integration, which promises its application as cloud-edge processors in internet-of-things networks (Supplementary Fig. 13).

Conclusions

We report a 3D circuit with eight layers of memristors for complex neural networks. To illustrate the capabilities of the approach, we programmed convolutional kernels into the 3D array for an accurate MNIST classification and effective edge detection in videos. Despite intrinsic device variations, we achieved a software-comparable performance and, moreover, a pixel-wise parallel-operation capability. The topologies of memristor arrays become important in hardware implementations of large DNNs. Employing the design in 3D can make this resource-expensive task into a manageable size, and provides substantial improvement to the speed and energy efficiency while running complex neural network models.

A broad range of applications can be envisioned with further optimizations in the device performance and more carefully designed on-chip peripheral analogue circuitry, even though the current 3D design might not be the most scalable architecture compared to a 2D counterpart. Furthermore, heterogeneous integration of our 3D structure with other functional circuit modules (such as photodetectors) will better take advantage of this unique monolithic integration technology and thus open up new opportunities in emerging hardware architecture for computer vision and brain-inspired computing[22].

Methods

Fabrication of 3D memristor arrays. The 3D array was fabricated on top of a silicon wafer with a 100-nm thick thermally grown oxide layer. Ti/Au contact vias were first made prior to the 3D integration. Additional fan outs were extended from the vias to provide I/O interfaces to the off-chip measurement system. The 3D integration process is schematically illustrated in Supplementary Figs. 4 and 5 for nano- and microscale memristors, respectively. The eight layers of electrodes were fabricated in sequence, with a SiO_2 passivation layer in between. For the nanoscale pattern, a HfO_2 switching layer and Ta input (top) electrodes were made in each layer; for the microscale pattern, deep vias were created to make contact with all the output (bottom) electrodes in each device layer, followed by switching layer and top electrodes fabrication at once for all eight layers. A 3.5-nm HfO_2 switching layer was deposited using atomic layer deposition at 250 °C followed by patterning and deposition of the Ta/Pt top electrodes. The 3D array fabrication was concluded after fabrication of the fan-out structures for the top electrodes.

Measurement system. The functionality of the 3D array was tested using a custom-built measurement system that can simultaneously apply voltage biases to 32 input channels and measure currents from 16 output channels (Supplementary Fig. 10). Details of the peripheral circuits design are described in previous reports[9]. The fabricated chip was mounted and wire bonded to a 44-pin chip carrier. Additional external circuits were built to operate the 3D array chips and demonstrate computing applications. Finally, to apply current compliance during programming, a separate switching matrix was inserted into the measurement system and controlled by a separate microcontroller. The switching matrix can provide an optional serial connection of a 5 kΩ resistor as an external current compliance.

Weight tuning in the 3D array. The conductance of each memristor was fine-tuned using a closed-loop algorithm that utilized the reliable analogue switching from our memristor. Each 3D row bank was operated individually. The program was capable of applying a train of analogue amplitude voltage pulses to one or multiple I/O ports to tune the conductance of selected memristors. The input and output channels of selected cells were biased at V_{program} and ground, respectively, where the input and output channels of unselected cells were biased at 1/3 V_{program} and 2/3 V_{program}, respectively (the so-called 1/3 voltage bias scheme). The conductance of each memristor was read out column by column by an automatic program inside the microcontroller to generate a 2D conductance map of each 3D row bank. The automatic tuning program with write/read cycles was repeatedly used until the desired conductance state was reached.

Parallel kernel assignment. There are different approaches to program the 3D array, as well as to utilize it for parallel processing. In this work, we designed an algorithm, as a proof-of-concept demonstration, to assign four filters and kernels repeatedly to the 3D array for both pixel- and filter-wise parallel processing (Supplementary Fig. 11). The filter-wise parallel processing could also be done at different row banks, but would require more row banks. The weight assignment in each 3D row bank is identical and calculated by an automated software program in MATLAB, and thus does not require excessive computing resources. Each row bank of the array handles one row of the kernels (1×3) and the outputs from three 3D row banks are combined to produce the final kernel output. Thereby, it is advantageous to process the 2D input (3×3) without the need of unrolling to a 1×9 vector in conventional 2D arrays. Two memristors are used to represent a signed synaptic weight and thus expand the kernel vector in each row to a (1×6) vector. In our 3D array, each staircase bottom electrode is used as the output and is connected to eight input electrodes through the memristor cells. The eight memristor cells provide the space to host the (1×6) kernel vector, ideally to be fit into the first six memristor cells (for example, when hosting one filter per row bank). In the meantime, adjacent output channels are used to program parallel operated kernels for different input pixels or different filters. However, each input channel is shared by a few localized output channels, and thus, once assigned to a particular output, require the later output channel and its kernel vector to comply with the existing pixel value in the input channels. Therefore, to decide the weight location of a new kernel output, the automated program may need to change the sequence of the kernel vector to match the existing input pixel values, which includes occasionally occupying the 7th and 8th cells in the output path (in the 7th and 8th layers of the 3D row bank). In some cases, failure to assign the (1×6) vector into a particular output path with eight cells because of the conflict with the existing input pixel values could also occur. In this case, the automated program will skip the current output column. The program will carry out iterations until it reaches the last output column of the 3D array. An example of the weight assignment process is shown in Supplementary Fig. 11b.

Data flow in the hybrid hardware/software CNN. As a proof-of-concept demonstration, only the parallel kernel operations were done in hardware (the 3D array), whereas the rest of the network components, which include activation functions, pooling layers and fully connected layers were implemented in software. In our experiments, inputs from the binarized MNIST digits were mapped to 0 V ('0') and 0.2 V ('1') and fed to the 3D array to start the convolution process. The outputs of the 3D array were measured by an in-house peripheral system and sent to a computer for further processing. The current level of the output feature was normalized to a numerical value by 0.2 mA, deducted from 0.2 V ('1' input) × 1 mS ('1' state) = 0.2 mA. Owing to the limited number of I/O channels from the measurement system, each row bank was selected and operated one by one from the external measurement system. Partial convolution results from each row bank were merged in software to produce the output of a convolution with an optional quantization step (round the numeric output to integer). The rest of the processes were also done in software.

Video processing in the 3D array. Two Prewitt kernels were programmed in the 3D array using the weight-tuning program described earlier. The dual 3×3 kernels were assigned using the same algorithm for CNN, as described above. The filter output current from the 3D array was converted and normalized to a numerical value between (0,1) in MATLAB and used to produce the final output in software. Compared with an ideal software output, the hardware-processed video frames were a little darker in the background (Supplementary Video 1). This was because of the noises and variations in the memristors that produced a non-zero output from the centre pixels. In ideal software output, the centre pixels were always discarded by the '0' weight to maximize the contrast in the edges. However, in any analogue system, such ideal zeros were very hard to obtain because of the finite signal-to-noise-ratio. Precise programming of the weights with differential output would provide a software-comparable quality, but further optimization of the device and system properties would be needed.

Data availability

The data that support the plots within this article and other findings of this study are available from the corresponding author upon reasonable request.

Code availability

The code that support the plots within this article and other findings of this study is available at https://github.com/plin83/NE_3D_CNN. The code that supports the communication between the custom-built measurement system and the integrated chip is available from the corresponding author upon reasonable request.

Received: 4 December 2019; Accepted: 3 March 2020;
Published online: 13 April 2020

References

1. Lecun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
2. Pei, J. et al. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature* **572**, 106–111 (2019).
3. Jouppi, N. P. et al. In-datacenter performance analysis of a tensor processing unit. In *Proc. 44th Annual International Symposium on Computer Architecture*https://doi.org/10.1145/3079856.3080246 (ACM, 2017).
4. Sangwan, V. K. et al. Multi-terminal memtransistors from polycrystalline monolayer molybdenum disulfide. *Nature* **554**, 500–504 (2018).
5. Van De Burgt, Y. et al. A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing. *Nat. Mater.* **16**, 414–418 (2017).
6. Strukov, D. B., Snider, G. S., Stewart, D. R. & Williams, R. S. The missing memristor found. *Nature* **453**, 80–83 (2008).
7. Jo, S. H. et al. Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett.* **10**, 1297–1301 (2010).
8. Choi, S. et al. SiGe epitaxial memory for neuromorphic computing with reproducible high performance based on engineered dislocations. *Nat. Mater.* **17**, 335–340 (2018).
9. Li, C. et al. Analogue signal and image processing with large memristor crossbars. *Nat. Electron.* **1**, 52–59 (2018).
10. Prezioso, M. et al. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **521**, 61–64 (2015).
11. Indiveri, G., Linares-Barranco, B., Legenstein, R., Deligeorgis, G. & Prodromakis, T. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology* **24**, 384010 (2013).
12. Ambrogio, S. et al. Neuromorphic learning and recognition with one-transistor-one-resistor synapses and bistable metal oxide RRAM. *IEEE Trans. Electron. Dev.* **63**, 1508–1515 (2016).
13. Gokmen, T., Onen, M. & Haensch, W. Training deep convolutional neural networks with resistive cross-point devices. *Front. Neurosci.* **11**, 538 (2017).
14. Agarwal, S. et al. Achieving ideal accuracies in analog neuromorphic computing using periodic carry. In *2017 Symposium on VLSI Technology* T174–T175 (IEEE, 2017).
15. Ambrogio, S. et al. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* **558**, 60–67 (2018).
16. Yao, P. et al. Face classification using electronic synapses. *Nat. Commun.* **8**, 15199 (2017).
17. Cai, F. et al. A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations. *Nat. Electron.* **2**, 290–299 (2019).
18. Glorot, X., Bordes, A. & Bengio, Y. Deep Sparse Rectifier Neural Networks. In *Proc. 14th International Conference on Artificial Intelligence and Statistics* (eds Gordon, G., Dunson, D. & Dudík, M.) 315–323 (PMLR, 2011).
19. Yuste, R. From the neuron doctrine to neural networks. *Nat. Rev. Neurosci.* **16**, 487–497 (2015).
20. Tanaka, H. et al. Bit cost scalable technology with punch and plug process for ultra high density flash memory. In *2007 IEEE Symposium on VLSI Technology* 14–15 (IEEE, 2007).
21. Topol, A. W. et al. Three-dimensional integrated circuits. *IBM J. Res. Dev.* **50**, 491–506 (2006).
22. Shulaker, M. M. et al. Three-dimensional integration of nanotechnologies for computing and data storage on a single chip. *Nature* **547**, 74–78 (2017).
23. Adam, G. C. et al. 3-D memristor crossbars for analog and neuromorphic computing applications. *IEEE Trans. Electron. Dev.* **64**, 312–318 (2017).
24. Luo, Q. et al. 8-Layers 3D vertical RRAM with excellent scalability towards storage class memory applications. In *2017 IEEE International Electron Devices Meeting, IEDM* 2.7.1–2.7.4 (IEEE, 2018).

25. Li, H. et al. Four-layer 3D vertical RRAM integrated with FinFET as a versatile computing unit for brain-inspired cognitive information processing. In *2016 IEEE Symposium on VLSI Technology* 1–2 (IEEE, 2016).
26. Li, Z., Chen, P. Y., Xu, H. & Yu, S. Design of ternary neural network with 3-D vertical RRAM array. *IEEE Trans. Electron. Dev.* **64**, 2721–2727 (2017).
27. Silver, D. et al. Mastering the game of go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
28. Strukov, D. B. & Williams, R. S. Four-dimensional address topology for circuits with stacked multilayer crossbar arrays. *Proc. Natl Acad. Sci. USA* **106**, 20155–20158 (2009).
29. Seok, J. Y. et al. A review of three-dimensional resistive switching cross-bar array memories from the integration and materials property points of view. *Adv. Funct. Mater.* **24**, 5316–5339 (2014).
30. Jiang, H. et al. Sub-10 nm Ta channel responsible for superior performance of a HfO_2 memristor. *Sci. Rep.* **6**, 28525 (2016).
31. Chen, W. H. et al. A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors. In *2018 IEEE International Solid-State Circuits Conference, ISSCC* 494–496 (IEEE, 2018).
32. Gao, L., Chen, P. Y. & Yu, S. Demonstration of convolution kernel operation on resistive cross-point array. *IEEE Electron. Dev. Lett.* **37**, 870–873 (2016).

Acknowledgements

This work is partially sponsored by the Air Force Research Laboratory under agreement nos FA8750-15-2-0044 and FA8750-18-2-0122, as well as by the Air Force Office of Scientific Research (AFOSR) under grant FA9550-12-1-0038. Q.X. thanks H. Weinstock of AFOSR for the initial support of this project. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the US Government. This work was performed in part at the Center for Hierarchical Manufacturing (CHM), an NSF-sponsored Nanoscale Science and Engineering Center (NSEC) at the University of Massachusetts, Amherst, and in part at the Center for Nanoscale Systems (CNS), a member of the NSF National Nanotechnology Infrastructure Network (NNIN) at Harvard University (ECS-0335765).

Author contributions

Q.X., J.J.Y. and P.L. conceived the concept and designed the experiments. P.L. did the fabrication, programming, measurements, data analysis and simulation. H.J. contributed to the device optimizations. Z.W. contributed to the 3D-1T1R measurements. C.L. and P.L. did the focused ion beam SEM and took the SEM images. C.L. contributed to the simulation. H.J., M.R., Y.Z. and N.K.U. helped with the fabrication. Y.L., P.L., W.S., Z.W. and C.L. built the measurement system and firmware. Q.X., J.J.Y. and P.L. wrote the manuscript. M.B. and Q.W. and all the other authors contributed to the results analysis and commented on the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41928-020-0397-9.

Correspondence and requests for materials should be addressed to J.J.Y. or Q.X.

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