nature electronics

Perspective

https://doi.org/10.1038/s41928-023-00977-1

Materials and devices as solutions to computational problems in machine learning

Received: 17 November 2021

Accepted: 12 May 2023

Published online: 26 July 2023

Check for updates

Nathaniel Joseph Tye[1,2], Stephan Hofmann[1] & Phillip Stanley-Marbell[1]

The growth of machine learning, combined with the approaching limits of conventional digital computing, are driving a search for alternative and complementary forms of computation, but few novel devices have been adopted by mainstream computing systems. The development of such computer technology requires advances in both computational devices and computer architectures. However, a disconnect exists between the device community and the computer architecture community, which limits progress. Here we explore this disconnect with a focus on machine learning hardware accelerators. We argue that the direct mapping of computational problems to materials and device properties provides a powerful route forwards. We examine novel materials and devices that have been successfully applied as solutions to computational problems: non-volatile memories for matrix-vector multiplication, magnetic tunnel junctions for stochastic computing and resistive memory for reconfigurable logic. We also propose metrics to facilitate comparisons between different solutions to machine learning tasks and highlight applications where novel materials and devices could potentially be of use.

For over 50 years, silicon has been the dominant material in computing devices. The success of silicon and silicon-based metal–oxide–semiconductor field-effect transistors (MOSFETs) in particular is due to a number of attributes: stable oxide formation (silicon forms a stable native oxide layer); abundance (silicon is the second most abundant element in Earth's crust[1]); miniaturization and mass production (MOSFETs can be miniaturized and mass-produced[2]); and power efficiency (complementary metal–oxide–semiconductor (CMOS) technology combines n-type and p-type MOSFETs into a single device that consumes power only when the device switches). The resulting digital computing devices must then have a number of general features[3]: gain (high gain in each component reduces issues arising from interdevice variation by using reference signal levels throughout a system to prevent signal degradation); input and output isolation (inputs and outputs must be isolated from each another to ensure that calculations are carried out in a predetermined manner); comparable on and off switching times (the switching time between the on and off states must also be comparable, otherwise a separate reset operation is needed, adding time and material costs); and inversion (a computer device must be able to convert a one to a zero and vice versa).

The rise of new application classes, such as machine learning (ML) is pushing the limits of conventional digital computing. For instance, ML training compute demands have doubled every two months since 2019[4]. This has typically been addressed by using larger numbers of processors such as graphics processing units (GPUs). However, this is an inefficient and unsustainable approach. As a result, there is an active search for alternative and complementary forms of computation. But despite considerable research effort, few novel devices have been adopted in mainstream computing systems.

ML is an approach to computation in which the computational process is learned from data. A dataset with one or more features (variables) has an associated output. ML is the attempt to discover

[1]Department of Engineering, University of Cambridge, Cambridge, UK. [2]Cambridge Graphene Centre, University of Cambridge, Cambridge, UK.
e-mail: phillip.stanley-marbell@eng.cam.ac.uk

the mathematical function that depends on these features and most accurately predicts the output. There are three key elements in ML methods: representation, evaluation and optimization[5–7]. In short, representations are the ML model, evaluations are a cost function or error metric, and optimizations are a method to determine the best representation.

ML methods and ML algorithms are not the same thing. For example, an ML implementation that uses a neural network as its representation, mean squared error as its evaluation and greedy search as its optimization will be made up of several algorithms (see Table 1 for examples). Most ML hardware accelerators focus on representations. For example, using novel devices to represent artificial neural network (ANN) neurons. However, other components of an ML algorithm could be accelerated as well. (See Supplementary Section 1 for a more detailed overview of ML, and Supplementary Section 2 for an overview of hardware accelerators for different ML representations.)

The development of computer technology for ML requires advances in both computational devices and computer architectures. However, a disconnect exists between the device community and the computer architecture community that poses a considerable hurdle to progress. In this Perspective, we explore the disconnect between communities with a focus on ML hardware accelerators. We argue that the direct mapping of computational problems to materials and device properties provides a route forwards (Fig. 1) and we examine novel materials and devices that have been successfully applied as solutions to computational problems (non-volatile memories for matrix-vector multiplication (MVM), magnetic tunnel junctions for stochastic computing and resistive memory for reconfigurable logic). We then propose metrics to facilitate comparisons between different solutions to ML tasks and highlight applications where novel devices and materials could potentially be of use.

ML accelerators

Many ML accelerators are described as neuromorphic, as their design is influenced, in part, by biological systems. However, the term neuromorphic is not well defined. For example, a hardware realization of a deep neural network (DNN) using logic gates to perform MVM might be referred to as a neuromorphic system, despite possessing no real resemblance to the brain. However, a system with artificial neurons and synapses might be referred to as neuromorphic. These systems include IBM's TrueNorth[8] (which features neurons and synapses), Intel's Loihi[9] (which features synapses, dendrites and axons) and Tianjic[10] (which features axons, synapses, dendrites and somas).

Attempts to simulate the brain, such as SpiNNaker[11], may also be considered neuromorphic, although SpiNNaker uses large arrays of processors, corresponding to a much larger area and energy consumption than the brain. Currently, the density of neurons and synapses in the brain far exceeds that of electronics technology, but suitable applications of materials and devices may close this gap. Our current understanding of the brain and animal nervous systems is also relatively limited, and much of the modern device literature relies on the neural network model of McCulloch and Pitts[12], and the Hodgkin–Huxley model of spiking neurons[13]. Although these have proven to be successful for neural networks, there are a number of more recent and more accurate neuron models[14–16].

We suggest that 'neuromorphic' be used exclusively to refer to systems that emulate biological components, rather than a general term for ML accelerators. We also suggest that neuromorphic hardware be categorized into two broad domains: systems that draw inspiration from biology and perform computations (such TrueNorth, Loihi and Tianjic), which are termed bio-inspired; and systems that simulate the brain (such as SpiNNaker), which are termed bio-mimetic. Such a distinction allows for a clearer understanding of a given system's purpose.

Neuromorphic accelerators provide a good example of inter-domain collaboration. For example, recent work highlights the opportunities for neuromorphic algorithms, namely, those that use spiking neural networks (SNNs)[17]. These opportunities fall into two broad categories of algorithmic approaches: those for ML applications and those for non-ML applications. In each case, the algorithms use an SNN, but ML applications require a training process whereas the non-ML applications use a hand-constructed SNN. The structure of an SNN is a directed graph and so these hardware SNNs can accelerate computations for graphs.

Neuromorphic hardware can also perform random-walk computations, with discrete-time Markov chains implemented on Intel's Loihi and IBM's TrueNorth platforms[18]. In addition, the development of perovskite-based devices that can be reconfigured to function as neurons, synapses, resistors or capacitors may prove to be particularly useful[19]; neurons and synapses are key components of hardware SNNs, and being able to selectively configure circuit elements to implement a specific algorithm or ML architecture could have considerable impact.

Table 1 | The three components of machine learning methods

Representation	Evaluation	Optimization
Instances *k*-nearest neighbour Support vector machines	Error rate	**Combinatorial** Greedy search Beam search Branch-and-bound
	Precision and recall	
Hyperplanes Naive Bayes Logistic regression	Squared error	**Continuous** Gradient descent Quasi-Newton methods Conjugate gradient Linear programming Quadratic programming
	Likelihood	
Decision trees	Posterior probability	
Neural networks	Information gain	
Graphical models Bayesian networks Conditional random fields	Kullback–Liebler divergence	
	Cost/utility	
	Margin	

Information adapted from ref. 6.

Points of disconnect

We identify disconnections between the device community and the computer architecture community in four areas: ML hardware versus software; the decline of CMOS; in-memory architectures; and ML workloads and Amdahl's law.

ML hardware versus software

The Modified National Institute of Standards and Technology (MNIST) optical character recognition task[21] is a common benchmark in ML and involves classification of handwritten digits between zero and nine. The MNIST dataset consists of 60,000 training examples and 10,000 test examples of handwritten digits, where each example is a 28 × 28 pixel greyscale image. Figure 2c shows the accuracies of several digital and silicon CMOS-based accelerators and approaches based on novel materials and devices against a given year. (Supplementary Tables 1 and 2 provide the data in this figure.)

The low accuracy for approaches based on novel materials and devices when compared with those based on digital silicon- and CMOS-based hardware (including software-programmable processors and GPUs), or fixed-function application-specific integrated circuits (ASICs) and other accelerators, is immediately apparent. ASIC accuracies lag software approaches by about 10 years, despite ASICs being a mature technology. ASIC and programmable hardware approaches both show a trend of progress, becoming more accurate over time. This, however, is not true for approaches based on resistive random-access memories (RRAMs) or other experimental devices. The 'other' approaches category is not a unified technology, but rather a range of different implementations. Thus a comparison is of limited value. These are included here instead to indicate the accuracies reported for approaches outside the most popular ones.

https://doi.org/10.1038/s41928-023-00977-1

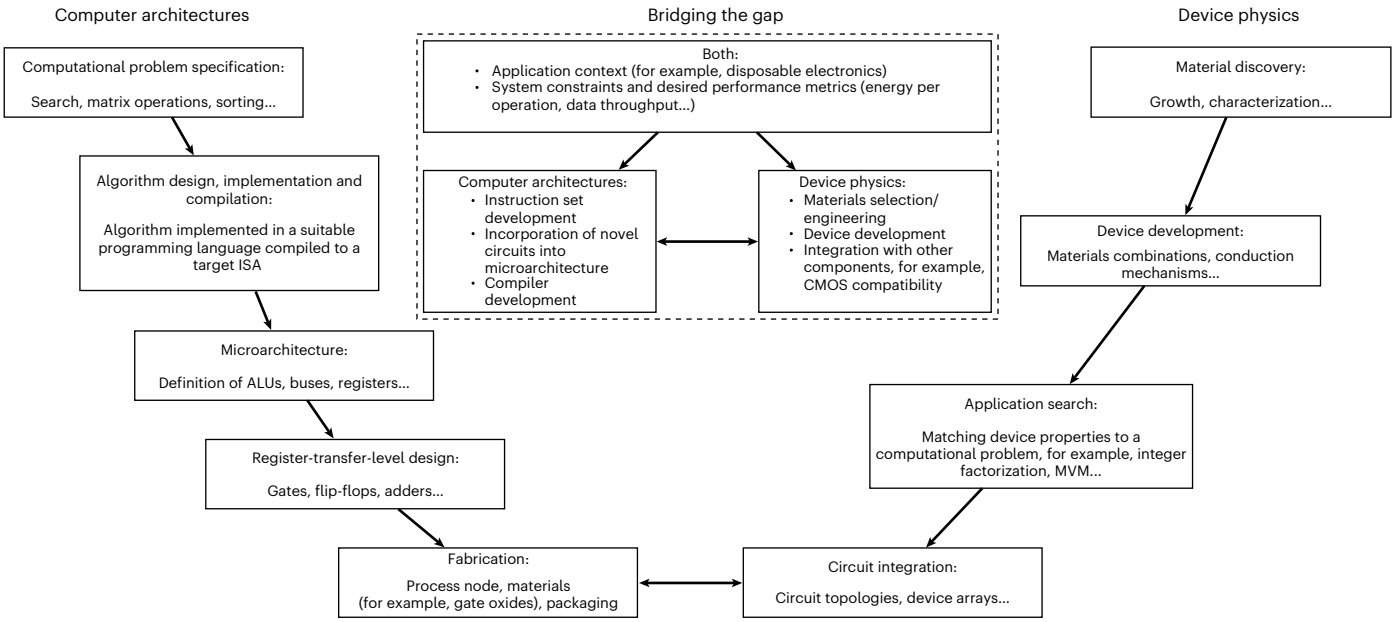


Fig. 1 | Development process for an integrated circuit from the perspective of a computer architecture and a device physicist. For a computer architect, the process typically begins by identifying a specific computational problem[79], an implementation of a specific algorithm and then a survey of existing technologies to arrive at a given implementation; in essence, a top-down approach. For a device physicist, the process begins with materials discovery, followed by device fabrication and then integration of devices into a larger circuit; a bottom-up approach, or a solution looking for a problem. The two processes currently intersect only at the end (fabrication and circuit integration). By intersecting earlier and by mapping computational problems to devices and materials, each step in each community can be brought together, creating more impactful research. ISA, instruction set architecture.

Although early work with RRAM dates back to 1965[22], progress was slow for several decades owing to the success of charge-storage memories. Thus, it is a relatively youthful technology compared with CMOS, fixed-function accelerators and software. Despite a renewed research interest in the past decade, there is no clear trend of increasing accuracy over time. The scarcity of full-scale physical realizations of such systems is a noted issue[23,24], but remains an important step. These systems are difficult engineering challenges, and would benefit from research collaboration at each level of abstraction. As well as scalable fabrication developments for existing devices, research into novel devices where large-scale manufacturing is not necessary could also enable more immediate application of novel technologies.

It is not uncommon in device research articles to see claims that experimental accuracies of 90%, for example, are comparable to a simulated hardware accuracy of 94%. Initially, this might appear relatively close. However, when the difference between the best and the tenth-best digital approaches is only 0.09 percentage points, a 4.4 percentage point difference is almost 50 times larger. Of course, the raw numbers do not tell the entire story. Leading digital approaches often have millions of parameters. One of the top algorithms at the time of writing has 1,514,187 trainable parameters[25]. In comparison, implementations based on novel materials and devices may have only hundreds and thus can be expected to not perform as well. This makes it clear that better metrics are required.

The decline of CMOS

A common notion in the device literature is the decline of CMOS, and articles often appear to treat CMOS circuits as synonymous with digital circuits and the von Neumann architecture. This is not the case. The von Neumann architecture is a model of computation, whereas CMOS is a fabrication process. Furthermore, many novel devices, including RRAMs, can be fabricated using CMOS processes and some of the best-performing ML accelerators for MNIST, such as IBM's TrueNorth[8], use a CMOS process. This illustrates the importance of distinguishing between (digital) CMOS and the von Neumann architecture.

Many ML hardware accelerators exceed the performance of TrueNorth[26], including Google's tensor processing unit (TPU)[27], which is an accelerator for conventional ANNs and thus mainly accelerates linear algebra computations. There is, however, a strong trend of increasing power consumption being associated with higher performance. CMOS remains the most successful technology for computing devices, regardless of the computer architecture they implement. Simply recreating CMOS devices using novel materials, while maintaining the same fundamental way of doing computation, that is, keeping the same architecture, has limited impact[28], as few match or better CMOS.

It is too early to say whether, under alternative computing approaches, CMOS technology will remain dominant and so it remains worthwhile to investigate different technologies. A hybrid approach, where new materials and devices are integrated into CMOS processes is one possible avenue. Alternatively, technologies such as probabilistic bits (p-bits), where the fundamental nature of the computation relies on completely different device and material properties, mean it is likely that CMOS will not be the best technology for all applications.

It is also important to note that model selection is equally important. For digital approaches, the ML model is often the key research focus, whereas, in the device domain, the device is the research focus. ML accelerators based on novel devices and materials would probably see performance gains if they implement ML models that are specifically designed to harness their properties and structures.

In-memory architectures

Many novel computing devices and circuits are often referred to as in-memory computing. This is an idea that attempts to address the von Neumann bottleneck. This bottleneck is the limitation in computational throughput inherent to von Neumann computers due to the separation of memory and processing (in particular, the shared program and data memory bus), which means that, while fetching instructions, a von Neumann computer cannot fetch any data to perform computations. In-memory computing addresses this bottleneck by co-locating computation and memory. Here the memory devices themselves perform

https://doi.org/10.1038/s41928-023-00977-1

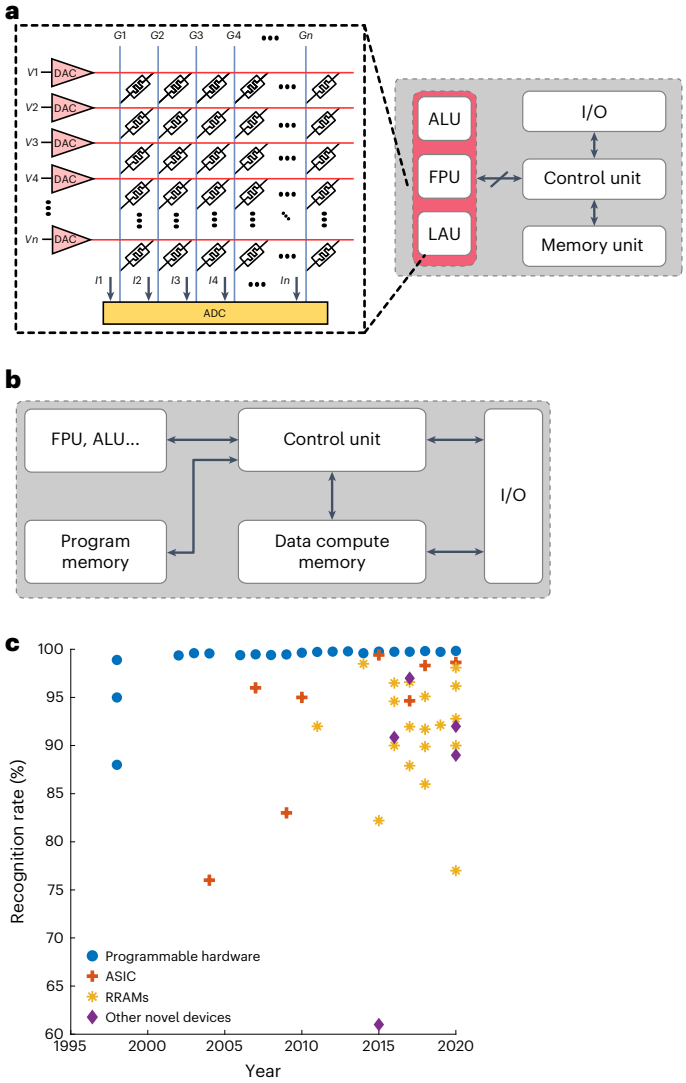


Fig. 2 | Hardware solutions to computational problems. a, A one-resistor RRAM crossbar array, and how such a circuit can be incorporated into a complete computer architecture, in this case, a von Neumann architecture. Alternative crossbar implementations may have a RRAM connected to a transistor (1T1R) to allow for individual device selection, or may use pairs of RRAMs to encode signed values[80]. ADC, analogue-to-digital converter; LAU, linear algebra unit; I/O, input–output. **b**, Schematic illustrating how an in-memory circuit may be incorporated into a complete computer architecture as data compute memory. In this case, the separate program and data memories make this a Harvard architecture. The control unit fetches and executes instructions from the program memory, including commands for in-memory computations to be performed, and can also fetch and send data to the FPU and ALU for processing. **c**, Scatter plot showing the reported accuracies on the MNIST dataset of several technologies currently being researched. Supplementary Tables 1 and 2 provide the models and details of these. *V*, input voltage; *G*, RRAM conductance; *I*, output current.

computations and so computation can occur in parallel with instruction access. However, the von Neumann architecture is not necessarily the dominant computer architecture in modern machines: the (modified) Harvard architecture is (Boxes 1 and 2; see also Supplementary Section 3 for an overview of computer architectures).

In practice, the majority of in-memory architectures are not computer architectures in the same sense as a von Neumann or Harvard architecture. Rather, they are circuits that perform a specific function, such as MVM. This makes them more akin to the arithmetic logic unit (ALU) or floating-point unit (FPU) in a hardware microarchitecture.

BOX 1

Definitions

Computational problem: a relationship between a set of inputs and outputs that defines some problem a computer can solve. For example, the problem of factorization takes some number as an input and has the factors of that number as an output.

Algorithm: a sequence of operations that transforms some input into an output, which is the solution to a computational problem, independent of implementation.

Computer architecture: an abstract description of a computer, such as an instruction set architecture, which defines functionality but not the physical implementation.

Computer device: an individual device used to represent data for computation, for example, a transistor, or a RRAM.

Software: an implementation of an algorithm or algorithms generally intended to be executed on reprogrammable digital CMOS hardware (for example, a microprocessor).

Neuromorphic system: a hardware device or circuit inspired by the behaviour of biological neural systems.

Fixed-function digital hardware: a device or circuit that processes data using digital logic in a fixed manner. For example, an H.264 decoder is an ASIC designed to decode video streams encoded in a common format.

Programmable architectures: a piece of hardware that can be reprogrammed for arbitrary functionality, such as a microprocessor. Such a system reuses the same logic elements for different applications.

Hardware microarchitecture: the physical specification of an instruction set architecture, defining ALUs, buses, registers and so on.

FPUs began as discrete components before being incorporated into conventional von Neumann or Harvard architectures, and the same may be true for in-memory approaches. Figure 2a illustrates how an MVM circuit could be incorporated into a complete computer architecture as a linear algebra unit. Because computer architectures require both program and data memory, the former to store instructions and the latter to store data, an in-memory architecture would still require circuitry for data input–output and storing and fetching instructions (Fig. 2b).

ML workloads and Amdahl's law

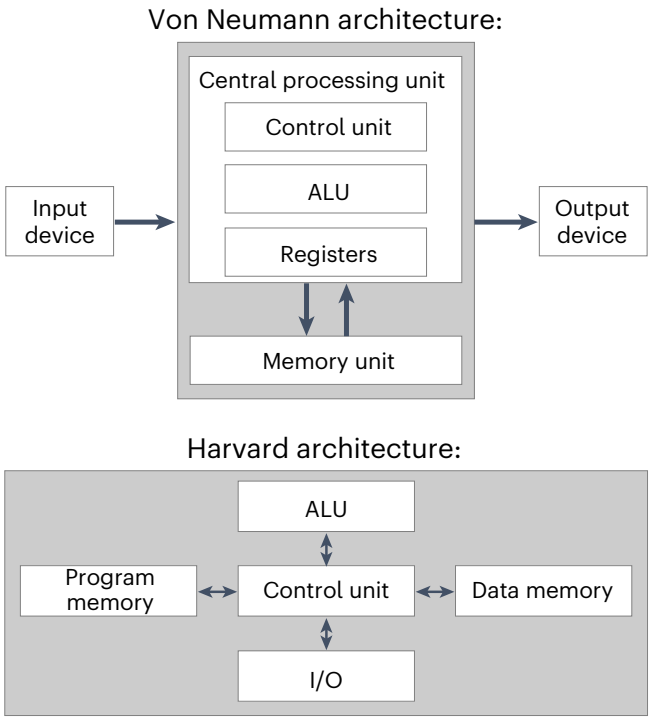
Workload characterization is an important part in understanding the computational demands of a given task. A breakdown of the execution time of eight contemporary DNNs implemented in TensorFlow, a standard ML framework (Table 2) provides a useful example of this[29]. The study investigates eight DNNs for different tasks, including sequence-to-sequence translation (text translation), end-to-end memory networks (natural language reasoning), deep speech (speech recognition), variational autoencoder (feature extraction, dimensionality reduction), residual networks (meta-learning), VGG-19 (computer vision), AlexNet (computer vision) and deep Q (deep reinforcement learning).

https://doi.org/10.1038/s41928-023-00977-1

BOX 2

Computer architectures

The von Neumann architecture stores program and data memory in the same place, whereas the Harvard architecture stores them separately. The main limitation of the von Neumann architecture is that, while loading an instruction from program memory, it cannot load or process data.



Direct TensorFlow matrix operations dominate the execution time in only two of the eight networks: deep speech (89%) and variational autoencoder (58%). For VGG-19, AlexNet, deep Q and residual networks, over 80% of the execution time was taken up by convolution operations, predominantly Conv2D, which is mainly a dot product operation. The other two convolution operations, Conv2DBackpropFilter and Conv2DBackpropInput are, however, not purely matrix operations, but also involve calculation of gradients, that is, differentiation.

Processing in-memory systems also exhibit similar trends[30], with convolution operations being the most computationally intensive across three ANN models (VGG-19, AlexNet and a deep convolutional generative adversarial network). This analysis also showed comparable operation times for in-memory and conventional central processing units (CPUs), with in-memory solutions being faster overall due to the reduced data transfer times. One interesting example is the text translation task. In this, over 50% of the execution time does not involve matrix operations, but rather element-wise and data movement operations. These characterizations only explore ANNs, however, and alternative ML methods may benefit from similar characterizations.

It is important here to also consider Amdahl's law[31]. This states that the performance gained by optimizing one part of a system is limited by the proportion of time the improved part is used. Even though an accelerator may improve the performance of an operation, such as MVM, if that operation is only a small part of the execution time of a method, then the overall performance improvement will be minimal.

Table 2 | Dominant TensorFlow operation categories in contemporary ML workloads

TensorFlow operation	Computational problems
Element-wise arithmetic	Addition, subtraction, multiplication, division, exponentiation, logistic function
Matrix operations	Matrix multiplication
Reduction and expansion	Logistic function, search, gradient, addition
Convolution	Matrix multiplication, differentiation
Random sampling	Non-uniform random number generation
Optimization	Moving average, differentiation
Data movement	Array operations

Dominant TensorFlow operation categories in contemporary ML workloads[29] and their underlying computational problems.

We also note that the most successful accelerators are fully integrated solutions, rather than a circuit performing a specific computation (Fig. 3c)[26]. This means their development involves input at multiple layers of abstraction, from the device level to the architecture level. Researchers at Google have, for instance, described the key considerations for the development of their v2 and v3 TPUs[27], where tasks were divided into two groups: those that need to be done well and those that can be done to a working level.

The tasks that must be done well consisted of: build quickly (fast design and fabrication of hardware, such as using readily available components and 'good enough' designs); achieve high performance (high-bandwidth buses and memory, systolic array structure for high computation density, use of the bfloat16 format, instruction-level parallelism); scale efficiently (ability to add multiple processors to cope with increasingly complex problems and datasets); easily adapt to new workloads (bfloat16 is easy for ML software to use, chip developed alongside compiler team to ensure programmability, the terminology of linear algebra used due to its generality); and be cost effective (the systolic array structure allows high density without demanding a large chip area, bfloat16 reduces hardware and energy costs, dual-core design, compiler-controlled memory hierarchy). This illustrates the importance of considering not just the device but also the compiler, the data precision and the instruction-level architecture.

Novel devices as alternatives to digital silicon electronics

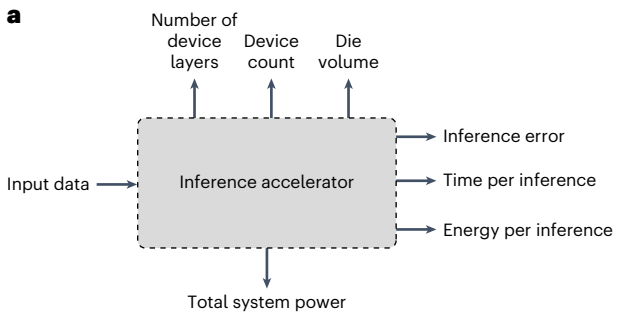
Traditional semiconductor roadmaps rely on the continuation of scaling laws and thus a key focus of device and materials research is exploring potential replacement to conventional silicon and digital CMOS. However, many novel technologies fail to compete with digital CMOS on metrics of energy, speed, scalability and price. However, this does not make them non-starters for computing applications. For example, in the analogue domain, the addition of two parallel 8-bit numbers requires only a single wire, using Kirchhoff's current law, but a digital CMOS circuit requires approximately 240 transistors[32]. Analogue multiplication requires 4 to 8 transistors, whereas digital multiplication demands as many as 3,000 transistors. Non-digital approaches can thus offer notable improvements in the efficiency and resource requirements of certain computations.

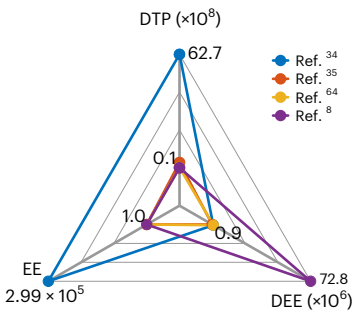
We highlight here three examples of such complementary technologies, where novel devices and materials are applied as solutions to computational problems. (See Supplementary Section 4 for an overview of novel devices as alternative to digital silicon electronics.)

Non-volatile memories for MVM

One example of mapping a computational problem to novel devices is the commonly explored use of non-volatile memories in crossbar arrays (Fig. 2a) to implement MVM operations. These use Ohm's law

https://doi.org/10.1038/s41928-023-00977-1





c

Technology	Device count	Die area (mm^2)	Number of layers	E_{inf} (J)	t_{inf} (s)	Error (%)	DTP (bits s^{-1})	DEE (% mm^{-2})	EE (%)	Ref.
OxRAM	7.72×10^4	0.0772	1	4.48×10^{-6}	1×10^{-6}	1.7	62.72×10^9	1.7×10^6	2.99×10^5	34
3D memristor	5.94×10^4	0.95	8	1.59×10^{-3}	2.12×10^{-5}	1.9	2.96×10^8	9.5×10^5	1.9	35
Memristor	2.62×10^3	0.563	1	1.87×10^{-5}	3.14×10^{-4}	3.81	2×10^7	1.77×10^6	30.5	64
CMOS	5.4×10^9	430	1	1.08×10^{-4}	1×10^{-3}	0.58	6.27×10^6	7.28×10^6	0.994	8

Fig. 3 | Metrics for evaluating a given solution to an ML task. a, Diagram showing a black-box accelerator and the corresponding input–output information for the metrics. **b**, Radar plot comparing four ML hardware accelerators[8,34,35,64]. **c**, Table showing the data from ML accelerators in the literature used to determine figures of merit. Each is for inference on the MNIST dataset[21]. The data size in each case is the number of pixels (784 in a 28 × 28 image) multiplied by the number of bits per pixel (8 bits in a 255-shade greyscale image), giving a total of 6,272 bits. For DTP, higher is better; for DEE and EE, lower is better. The OxRAM and memristor systems all use a 1T1R structure, whereas the digital system uses digital CMOS technology.

and Kirchhoff's law, where the sum of currents from each device gives a final output current. The value of the output current corresponds to a particular signal strength. The major attraction of this approach for computer architects is that it substantially reduces the complexity of the MVM operation.

Using the non-volatile memory crossbar approach, the entire operation in the analogue domain is completed in a single digital circuit's clock cycle, reducing the complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(1)$ (ref. 33), that is, constant time. (See Supplementary Section 5 for an overview of computational complexity theory and big-O notation.) In Supplementary Section 5, we note that an algorithm's complexity is independent of the hardware on which it is running, that is, an algorithm will remain $\mathcal{O}(n^2)$ regardless of implementation, and this remains the case here. MVM takes $\mathcal{O}(n^2)$ time in software running on a CPU and $\mathcal{O}(n)$ space in memory. In contrast, MVM takes $\mathcal{O}(1)$ time using a crossbar, but it requires $\mathcal{O}(n^2)$ devices to perform. The complexity thus transfers from time to space, as the crossbar performs the MVM operation, rather than the CPU.

A key advantage of non-volatile memories in the above example is that only one or two devices are required (for example, one memory or one transistor and one memory, depending on the system configuration), compared with potentially thousands using digital logic, meaning a substantial reduction in materials, and thus potential for higher-density circuits.

As an active field of research, non-volatile memory crossbars have been demonstrated for a range of ML implementations, including DNNs[34], convolutional neural networks (CNNs)[35,36] and SNNs[37–39]. However, non-volatile memory implementations face a number of well-established challenges, including interdevice variation, cycle endurance and parasitic wire resistance, which becomes an issue in the large arrays required by modern ML applications. These are engineering challenges that can be solved through research in multiple domains, with a large part of this being addressed by the materials community. For example, wire resistance can be reduced by three-dimensional (3D) integration[40], and physical mechanisms such as Mott transition RRAM can be exploited to reduce variability and improve endurance[41]. Avenues for improving RRAM performance include: interface engineering, element doping of functional materials and introduction of low-dimensional materials[42]. Non-volatile memory-based implementations of CNNs offer a solution to some of the scalability issues affecting fully connected solutions. In particular, they require fewer weights and thus fewer devices, reducing both the influence of parasitic resistances and the precision issues resulting from interdevice variation. Thus, non-volatile memory approaches would also benefit from suitably designed CNN architectures.

Magnetic tunnel junctions for stochastic computing

Although magnetic random-access memories (MRAMs) have been proposed as a potential 'universal memory'[43], they also have applications for computation. One particularly interesting application is probabilistic computing using magnetic tunnel junctions (MTJs) as p-bits. The term p-bit initially described a switch that stores a bit, either 1 or 0, by representing the bit as a microstate[44]. A microstate is a specific microscopic configuration that a thermodynamic system occupies with some probability. Contemporary work instead considers p-bits as a computational primitive analogous to bits in Boolean logic, or qubits in quantum computing[45]. Unlike bits, which are either 1 or 0 at a given moment, or qubits, which exist in a superposition of states, where they are both 1 and 0, p-bits rapidly flip between 1 and 0 and so have a probability of being 1 or 0 at a given moment. In essence, p-bits are a hardware implementation of a Bernoulli random variable. This property allows for them to directly perform computations on probabilities, athough p-bits also have quantum-inspired and ML applications[46]. For example, p-bits can implement adiabatic quantum computing algorithms to perform integer factorization[47], by considering it as an optimization problem.

Although not implementing p-bits specifically, strained magnetic tunnel junctions (S-MTJs) can be used for probabilistic computation[48]. In this approach, the magnetic domain of the S-MTJ represents a probability vector, with each digit representing a possible outcome of a

discrete random variable, as opposed to using MTJs as a p-bit. Compared with an implementation of the same Bayesian network using 5-bit digital CMOS multipliers, S-MTJ simulations suggest area reductions of up to 127 times, a 214-times power reduction and a 70-times lower latency.

The key difference between p-bits and MRAMs is that p-bits make use of a typically undesirable property of MTJs: the instability resulting from superparamagnetism. Superparamagnetism is the phenomenon whereby, under the influence of temperature, the magnetization of the magnet randomly flips direction. A high energy barrier (E_b) is desirable for regular MTJs as it increases the time taken for this flip to occur. For p-bits, where this flipping is desirable, a low- or zero-barrier magnet, where $E_b < k_BT$ (where k_B is Boltzmann's constant and T is temperature) is ideal, as the time period for a flip would be less than 1 ns (ref. 46). To the best of our knowledge, all p-bit implementations utilize a traditional MTJ structure with bulk materials, and so there is scope for impactful research in this area using two-dimensional materials. These are predicted to have very low magnetic anisotropy energies: the barrier of monolayer CrI_3 has been measured to be as low as 0.66 meV (ref. 49) corresponding to a transition time, τ_N, of about 1 ns. Another candidate material, strained Fe-doped MoS_2, has a barrier of about 1.3 meV (ref. 50), corresponding to a τ_N of 1.06 ns, decreasing with the addition of strain. For comparison, p-bit approaches using bulk materials give best-case retention times of a few milliseconds[47], with other simulations giving a time of 1.93 ms (ref. 51). S-MTJs can have retention times of a few nanoseconds[52–54]; however, these require an additional piezoelectric layer, adding material and space costs.

More generally, p-bits are simply tunable random number generators. Spintronics are just one possible implementation of p-bits: there may be inherent properties of other materials, that is, controlled stochastic processes, which would be equally useful or even better-suited to implementing p-bits. This illustrates well the intersection of devices, materials and computational research. As p-bits offer a fundamentally different way to do computations by physically representing probabilities, they map computational problems such as Bayesian networks directly to hardware, which is of interest to both computing and devices researchers. However, as their performance is directly tied to materials properties, the materials selection and engineering aspect links directly to the computational problem.

Resistive memories for reconfigurable logic

Although selection of suitable materials for a given application is important, materials design and engineering also offers an avenue for substantial impact. Materials design and development may follow a similar format to product design, serving as a natural interface between materials developers and system designers[55]. Likewise, 'intelligent matter'[56] provides another interesting opportunity. So-called intelligent matter falls into several classes: swarm-based, self-organized materials, soft-matter implementations and solid-matter implementations. The final category has been the primary area of exploration for computing applications, with reconfigurable devices being a growing area of research.

'Molecular memristors'[57], show how selective engineering of materials allows for computational functionality to be embedded directly into materials properties. The molecular memristor uses different redox states to represent different logic functions, including XNOR, NAND, NOR, AND and OR on up to four bits, as well as using an array of devices to implement a decision tree accelerator. The reconfigurable nature of the devices makes a circuit of them something analogous to a field-programmable gate array but with a potentially smaller device footprint. As the devices implement digital logic functionality, their performance compared with conventional CMOS is of interest. As the memristors implement logic functions directly in a single device, this means fewer devices are needed for a given logic gate, reducing the hardware and materials footprint. Another attractive prospect of this approach is that the devices have a high cycle endurance (about 10^{10}), as well as a very low switching energy (360 aJ). The authors note that, given the intended application of these devices, one must consider also the energy required to move data, and so system-level metrics are more sensible for comparison, an idea we discuss in 'Metrics, applications, and avenues of collaboration'. Given the digital nature of these devices, it is also worthwhile to consider how they match Keyes' criteria[3]. The presented devices are passive and so do not have gain, nor do they appear to have input–output isolation, although the incorporation of a transistor for a one transistor, one resistive memory (1T1R) structure might aid with both of these. The devices do, however, appear to have comparable on–off switching times.

Table 3 | Common activation functions and associated (worst case) complexities for ANNs

Function	Equation	Complexity
Identity	$f(x)=x$	$\mathcal{O}(n)$
Binary step	$f(x)=\begin{cases}0, & x<0\\1, & x\geq 0\end{cases}$	$\mathcal{O}(1)$
Tanh	$f(x)=\tanh(x)$	$\mathcal{O}(M(n)\log(n))^a$
ReLU	$f(x)=\begin{cases}0, & x<0\\x, & x\geq 0\end{cases}$	$\mathcal{O}(n)$
Sigmoid	$f(x)=\frac{1}{1+e^{-x}}$	$\mathcal{O}(M(n)\log(n))^a$
ArcTan	$f(x)\tan^{-1}(x)$	$\mathcal{O}(M(n)\log(n))^a$

$M(n)$ represents the complexity of a given multiplication algorithm. ReLU, rectified linear unit. [a]Using arithmetic-geometric mean iteration[20].

An interesting application of reconfigurable devices would be hardware implementation of activation functions. For example, 30% to 40% of the execution time for machine translation using GPUs is taken up by the final SoftMax activation layer[58]. Activation functions are a key part of ANNs, as they introduce nonlinearities to ANNs, which is vital for deep networks. In SNNs, the activation function acts as a threshold, determining whether a given neuron should fire. Table 3 shows the most common activation functions and their complexities. If the wrong activation function is chosen for a given ANN hardware implementation, then it can severely limit the performance. An implementation of activation functions that directly exploits material or device properties may then offer substantial advantages. If some novel device implements an activation function in the analogue domain and operates in a single digital clock cycle, then the activation functions could be reduced to a complexity of $\mathcal{O}(1)$.

Some work exists in this domain already[59,60]; however, there does not yet seem to be a notable focus in this area, despite the potential impact. For example, machine vision integrating sensing and ANN processing[61] can give reduced hardware footprints, although off-chip implementation of the activation function introduces communication and hardware overheads. A fully integrated solution would further improve the impact of such an approach. Suitably designed reconfigurable devices may then simultaneously reduce the hardware footprint while enabling more general circuitry in this domain.

Metrics

Establishing standard figures of merit to evaluate and compare approaches is important and some work on benchmarks for ML accelerators already exists[26,62]. These discuss ideas such as quantifying the energy breakdown for an in-memory accelerator based on SRAM and RRAM. Inference and training also have different demands. For example, almost 68% of the energy during training is used for DRAM access[62]. Constrained and unconstrained die areas also have an influence, with 22 nm RRAM-based implementations showing that larger dies have a lower total energy consumption, at the cost of material

requirements. For example, a 228 mm^2 die for inference requires about 400 μJ of energy, compared with almost 600 μJ for a 6.5 mm^2 die. For space-constrained applications, a 228 mm^2 die may be unfeasible, and this serves as a good example of the trade-offs that are necessary when considering novel devices as solutions to computational problems. This work serves as a useful reference when considering the specific technologies for an RRAM in-memory accelerator. In contrast, the following metrics we propose are intended as system-level measures, applicable not just to ANN accelerators but also other accelerators, such as support vector machines (SVMs) or decision trees.

For standard benchmarks such as MNIST, simply reporting the classification accuracy is not a good measure, as the best-performing digital approaches utilize millions of parameters and thus billions of transistors and clock cycles. When compared with an implementation based on novel materials with only a few hundred devices, the comparison becomes unfair. In keeping with the perspective of viewing novel devices, materials and circuits as solutions to computational problems, we suggest that it is better to evaluate systems as black boxes, where the relations between inputs and outputs are used as performance indicators. We consider a general accelerator as a black box with the following output information available (Fig. 3a): input data size (number of bits; the number of bits in the input data for a single inference, for example, the size of a single image in bits); device count (the number of devices, that is, the total number of devices (for example, transistors, RRAMs, p-bits and so on) in a system); die area (A; the die area in mm^2); the number of layers (number of layers) (the number of device layers for 3D-integrated dies); total system power (TSP; the total power consumed by an accelerator); inference energy (E_{inf}; the total number of joules required to perform a single inference or classification); inference time (t_{inf}; the time taken in seconds to perform a single inference); data size (the size of the input and output data in bits or bytes); error (error on a given task as a percentage, for example, the percentage of incorrectly classified digits in MNIST).

Data throughput

Floating-point operations per second (FLOPS) is a common metric for CPUs and may be useful for ML accelerators if they operate on floating-point numbers. However, not all ML systems do. For example, binary ANNs operate on Boolean values, or an ANN may only accept integers. OPS rather than frames per second may make a better benchmark[62] as this is dataset dependent and suited in particular to computer vision tasks. We instead propose that a better metric is the data throughput (DTP) of a system, that is, how many bits of data an accelerator can process in a second. A metric such as multiply-accumulate (MAC) operations per second would only applicable to accelerators where MVM is the main operation. For systems such as decision trees, this metric does not make sense. Likewise, systems using dynamic vision sensors process spike events, rather than frames[63] and so frames per second is not a useful metric. DTP is therefore platform agnostic and is applicable for both training and inference. We define the DTP as:

$$\mathrm{DTP} = \frac{\text{Number of bits}}{t_{\mathrm{inf}}}. \tag{1}$$

The issue of dataset-dependent measures does not apply in this case, as we intend our metrics be used to evaluate how a given accelerator performs on a given computational problem compared with others, that is, to evaluate like-for-like tasks.

Error efficiency

Many high-performance ANNs have over a million different parameters and many layers. This corresponds to a high cost in time, resources and energy consumption. By comparison, approaches based on novel materials and devices, such as RRAM crossbars, use substantially fewer resources; however, they generally have lower accuracies. Ideally, the higher the accuracy the better, but the time and power costs can offset the advantages of accuracy. For example, a facial recognition algorithm that unlocks a mobile phone only when it is 99.98% certain the right face has been scanned is not much good if it takes several minutes to run and severely depletes the battery. However, an ML algorithm used for medical diagnosis should be as accurate as possible. We suggest that the error efficiency (EE), which relates the wasted power (the product of the error on a given task and the power for a single inference) to the total system power, that is, the overall power consumption of the accelerator, is a useful metric to evaluate devices. For example, in low-power environments, such as edge applications, larger CMOS feature sizes are used due to their lower leakage currents. However, these have larger switching energies than smaller processes, and so the energy for a single inference may be larger. Thus, the wasted power may be a larger proportion of the total system power. If a novel device was instead used, with much smaller power draw per inference but lower accuracy, then a lower accuracy may be compensated for by the improved energy efficiency. We define the EE as follows:

$$\mathrm{EE} = \frac{\mathrm{Error} \times \frac{E_{\mathrm{inf}}}{t_{\mathrm{inf}}}}{\mathrm{TSP}}. \tag{2}$$

Device error efficiency

Following from our proposal for EE as a metric, we propose a similar metric for the device count: device error efficiency (DEE). In effect, DEE looks at the proportion of wasted devices for a given inference operation. This relates the error per inference to the device density, and is equal to the product of the error and the density of devices (that is, the number of devices in a given die area), and so also encapsulates process geometry. To account for accelerators that use 3D integration to improve device density, we multiply the device density by the number of layers. For non-3D systems, this value is one. In a system where material constraints are an issue, one may wish to trade accuracy for a lower device count. For example, a system with a low error but many devices may be less efficient per device than a system with a lower accuracy but far fewer devices, and so the latter may be more useful in material- or geometry-constrained environments. We thus define DEE as:

$$\mathrm{DEE} = \mathrm{Error} \times \text{Number of layers} \times \frac{\text{Number of devices}}{A} \tag{3}$$

Evaluation using metrics

As a demonstration of how our metrics can facilitate comparisons between different hardware accelerators, we present a comparison of several hardware accelerators from Fig. 2b. Figure 3c shows the data we use to calculate the figures of merit for the radar plot in Fig. 3b. We selected these to account for both different hardware implementations, for example, digital CMOS versus non-volatile memory, and different ML methods, for example, SNNs versus CNNs. As the data show, despite TrueNorth having the highest accuracy and device count, other approaches have a higher throughput. The metal–oxide resistive random-access memory (OxRAM)-based accelerator has the highest throughput, although it also has the largest DTP and better accuracy than the 3D-integrated memristor accelerator[35], its higher power consumption and lower device density means it is less competitive in EE and DEE. Owing to the greater device density enabled by 3D integration, the 3D-integrated memristor accelerator surpasses the others in DEE, despite the lowest reported accuracy. TrueNorth excels in error efficiency, with a very small amount of wasted energy compared with alternative approaches. These examples thus illustrate the value of our metrics, as simple measures of device count, power consumption and accuracy do not necessarily mean a given solution is best in this respect. Particularly in the DTP, we see that the 3D-integrated

memristor accelerator[35] is only a single order of magnitude faster than the standard memristor accelerator[64], despite the reported number of operations per second (OPS) being much larger (1,460.7 TOPS s^{-1} versus 81.92 GOPS s^{-1}).

The example above also illustrates the value of considering a given device or circuit as a solution to a computational problem. Consider our definition of a computational problem in Box 1. In this instance, we have as an input a set of images showing handwritten digits and our outputs are a set of numbers from zero to nine. The hardware accelerator is, in this case, a solution to this problem, transforming the input into a desired output. Our metrics thus illustrate how different solutions to the same problem perform better or worse, depending on the target use-case. For low-power applications, we see that an approach such as TrueNorth performs the best, whereas for pure throughput, the OxRAM-based accelerator performs best. By considering the metrics and parameters we propose, we suggest that novel devices, materials and circuits can be better applied as solutions to computational problems. One analysis of ML workloads discusses the necessary trade-off between energy consumption and processing time[30], particularly for in-memory approaches. Thus the 'best' performance is context specific.

Applications

Given the relatively poor performance of devices and systems based on novel materials when compared with conventional CMOS, application domains where the disadvantages become irrelevant or are minimal are of interest. Below, we suggest some example domains and device/materials requirements for these areas.

Disposable electronics

A growing area of interest is disposable electronics. These are circuits that are designed to be used a limited number of times before being thrown out. Examples may include medical sensors or packaging. Given their inherent limited lifespans, such circuits would not need high endurance and would also need to be very low cost. Thus, CMOS processes may be prohibitively expensive or unsuitable for such applications. There is also a large potential overlap with printed electronics in this domain and so the rigidity of CMOS devices may also be an issue. In this area, devices based on materials such as two-dimensional materials or polymers[65] would be ideal, due to their ease of manufacture and flexibility. Furthermore, their disposable nature means that the low cycle endurance of such devices would not be an issue.

Space electronics

The space sector has grown rapidly in recent years, with the number of scientific and commercial launches growing since 2004[66]. Given the usefulness of ML and hardware acceleration, it is clear that there is scope for ML hardware accelerators in the space sector. Such applications can be considered computing at the extreme edge; space probes, rovers and satellites operate on very low power, with very limited bandwidth for data transmission. Thus, pre-processing of gathered data, for example, via some edge ML application, may have a large impact on the information gained from such missions. However, space electronics have very specific requirements. In addition to low-power operation, the large amounts of radiation both in space places unique demands on circuits, in particular, their need for radiation hardness. Some work already exists exploring the radiation hardness of RRAMs[67,68] and a design flow for rad-hard non-volatile memories already exists[69]. For ML inference accelerators for space applications, materials and devices with a robustness to radiation would be essential. As space missions are often produced only once, scalable fabrication would also be less of an issue.

Biocompatible electronics

Medical implants are a growing field of research and industry and such devices also have specific requirements, biocompatibility being particularly important. For example some neural implant that interfaces with neurons to predict seizures would need to both operate on very low power and be biocompatible. Some examples of experiments studying biocompatibility on flexible electronics already exist[70], but further work would be of interest.

Table 4 | The 13 computational dwarfs identified by University of California, Berkeley researchers and their corresponding applications in ML

Dwarf	ML application
Dense linear algebra	SVMs, PCA, ICA
Sparse linear algebra	SVMs, PCA, ICA
Spectral methods	Spectral clustering
N-body methods	–
Structured grids	–
Unstructured grids	Belief propagation
MapReduce	Expectation maximization
Combinational logic	Hashing
Graph traversal	Bayesian networks, decision trees, natural language processing
Dynamic programming	Forwards–backwards, inside–outside, variable elimination, value iteration
Back-track and branch-and-bound	Kernel regression, constraint satisfaction, satisfiability
Graphical models	Hidden Markov models
Finite state machine	–

Quantum electronics

Quantum computing is a burgeoning field with a number of potential applications and quantum ML is a growing field of research. Many quantum computers operate at very low temperatures to minimize decoherence and reduce errors. Non-volatile memories may have utility here, and some work on quantum memristors already exists[71,72]. Given such devices would probably interface with quantum hardware, they would need to perform at low temperatures and so temperature-dependent measurements, as well as experiments to explore the amount of noise in different devices would be of interest.

Recurrent networks

Recurrent neural networks (RNNs) are another avenue where the metrics for a good device may be different. RNNs are a type of ANN that have a time dependence, making them suited to tasks using time series data, such as natural language processing[73]. RNNs use previous outputs as inputs, that is, the activation at a given time is a function of the activation at a previous time. Such networks may not require as many layers and so smaller arrays of devices may be suitable. However, given that RNNs may operate on many time steps, a high cycle endurance would be necessary. Reservoir computing is an ML approach that builds on several RNN models. The reservoir part of a reservoir computer is treated as a black box, but has two requirements: it must be made up of individual nonlinear units and be able to store information[74]. A reservoir computer uses the reservoir to map inputs into a higher-dimensional computing space and then conducts pattern analysis in a readout section. Unlike other ANN approaches, a reservoir computer does not train the weights of the input or reservoir sections, only the readout. This theoretically means a simplified and faster training process, as simple training algorithms, such as linear regression, can be used, with a focus on reduced computational cost compared with alternative approaches[75]. As with accelerators for other ML approaches, reservoir computers have been realized using different devices and materials, including a photonics-based approach[76], or even a literal reservoir in the form of a water tank[77].

Outlook

In 2006, researchers at University of California, Berkeley met to discuss and make predictions for the transition towards parallel computing[78]. They described 13 'dwarfs': algorithmic methods that describe patterns of data communication and computation. Not all the dwarfs are applicable to ML, and there is some overlap in applications (dense and sparse linear algebra both have utility for SVMs, principal component analysis (PCA) and independent components analysis (ICA), for example). These dwarfs pose an additional consideration. Conventional wisdom may suggest that increased parallelization always improves computation efficiency. However, one must also consider the movement of data itself, and the complexity of a given computation. The classification also accounts for ML applications. Table 4 lists the 13 dwarfs and their corresponding applications in ML.

Fundamentally, most ML applications are collections of more general computational problems. The properties of novel devices and materials may have the potential to perform these computations in different and more efficient ways than existing algorithms. If we consider the fundamental ML computations in isolation, ignoring the minutiae of a given ML application, we may find that novel devices have wider potential applications than simply implementing a given ML model. If we consider the fact that non-volatile memory crossbars implement MVM, rather than a neural network, we see that they also have utility as linear algebra accelerators, giving them application to the first two dwarfs.

Modern ML implementations in fixed-function hardware typically make use of the bfloat16 format, as it allows for fast conversion to 32-bit floating-point formats while reducing ML algorithm storage requirements and computation time. For a 7 nm process, bfloat16 offers a 1.5-times reduction in energy consumption compared with a conventional Institute of Electrical and Electronics Engineers 16-bit float format[27]. We thus propose that this be a common standard of precision used for ML systems.

We previously discussed the different understandings of the term 'computer architecture' between communities, and how a simple MVM circuit is not a computer architecture, but rather a hardware implementation of an algorithm. These different understandings of the term hamper collaboration, and greater precision is required, as not all ML accelerators are fully fledged computer architectures.

Intel's Loihi[9] is an example of an accelerator that can be considered a computer architecture in the strict definition of the term. Loihi's instruction set features common operations such as bitwise operations, comparisons (for example, less than, not equal), and basic arithmetic operations, but also includes specific instructions for spiking neural networks (SPIKE and PROBE). The former generates a spike and the latter sends probe data to a processor. This is another key point for early-stage collaboration between device physicists and computer architects: the development of new instruction sets. For their incorporation into CPU architectures, accelerators such as in-memory units will need corresponding instruction sets. Although the specific implementation may vary (for example, MRAM, RRAM and so on), a standardized instruction set means that a given implementation can account for these accelerators.

A final suggestion we propose is the standardized reporting of the computational cost in devices articles. This will not only help researchers consider the scalability and viability of their own research but also help readers and those who build on their work to better direct subsequent investigations. Returning to the example of non-volatile-memory-based MVM operations, we can see that changing the architecture reduces the time complexity of the operation, but, more importantly, this does not smuggle the complexity elsewhere: the resource complexity grows linearly with input size.

For ML approaches, whether based on digital logic implemented in CMOS, or on digital- or analogue-domain computation using novel materials and devices, the understanding that any ML algorithm has three components makes this easy to calculate, as the complexity of the system or algorithm will be whichever term in the learning equals representation plus evaluation plus optimization equation grows the fastest.

Many review articles in the device literature discuss emerging technologies and their principles of operation, but ultimately end with the message that useful implementations are a distant prospect due to engineering challenges and issues with scalability compared with current CMOS devices. We instead suggest that emerging technologies may be better applied and commercialized if researchers shift their focus to the direct mapping of computational problems to the unique properties of new devices to achieve better performance in a given application than conventional digital CMOS devices using the above metrics.

Data availability

All relevant data are included in the paper and/or its Supplementary Information files.

References

1. Rumble, J. & Bruno, T. *CRC Handbook of Chemistry and Physics 2019-2020: A Ready-reference Book of Chemical and Physical Data* CRC Handbook of Chemistry and Physics (Taylor & Francis Group, 2019).
2. Moskowitz, S. *Advanced Materials Innovation: Managing Global Technology in the 21st century* (Wiley, 2016).
3. Keyes, R. W. What makes a good computer device? *Science* **230**, 138–144 (1985).
4. Mehonic, A. & Kenyon, A. J. Brain-inspired computing needs a master plan. *Nature* **604**, 255–260 (2022).
5. Abu-Mostafa, Y. S., Magdon-Ismail, M. & Lin, H.-T. *Learning From Data* (AMLBook, 2012).
6. Domingos, P. A few useful things to know about machine learning. *Commun. ACM* **55**, 78–87 (2012).
7. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016).
8. Merolla, P. A. et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345**, 668–673 (2014).
9. Davies, M. et al. Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).
10. Pei, J. et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature* **572**, 106–111 (2019).
11. Painkras, E. et al. SpiNNaker: a 1-W 18-core system-on-chip for massively-parallel neural network simulation. *IEEE J. Solid State Circuits* **48**, 1943–1953 (2013).
12. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **5**, 115–133 (1943).
13. Hodgkin, A. L. & Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **117**, 500–544 (1952).
14. Ermentrout, G. B. & Kopell, N. Parabolic bursting in an excitable system coupled with a slow oscillation. *SIAM J. Appl. Math.* **46**, 233–253 (1986).
15. Jolivet, R., Rauch, A., Lüscher, H.-R. & Gerstner, W. Predicting spike timing of neocortical pyramidal neurons by simple threshold models. *J. Comput. Neurosci.* **21**, 35–49 (2006).
16. Galves, A. & Löcherbach, E. Infinite systems of interacting chains with memory of variable length—a stochastic model for biological neural nets. *J. Stat. Phys.* **151**, 896–921 (2013).
17. Schuman, C. D. et al. Opportunities for neuromorphic computing algorithms and applications. *Nat. Comput. Sci.* **2**, 10–19 (2022).
18. Smith, J. D. et al. Neuromorphic scaling advantages for energy-efficient random walk computations. *Nat. Electron.* **5**, 102–112 (2022).

19. Zhang, H.-T. et al. Reconfigurable perovskite nickelate electronics for artificial intelligence. *Science* **375**, 533–539 (2022).
20. Brent, R. P. *Multiple-precision Zero-finding Methods and the Complexity of Elementary Function Evaluation* 151–176 (Academic Press, 1976).
21. Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
22. Hiatt, W. R. & Hickmott, T. W. Bistable switching in niobium oxide diodes. *Appl. Phys. Lett.* **6**, 106–108 (1965).
23. Hu, M. et al. Memristor-based analog computation and neural network classification with a dot product engine. *Adv. Mater.* **30**, 1705914 (2018).
24. Li, C. et al. In-memory computing with memristor arrays. In *2018 IEEE International Memory Workshop* 1–4 (IEEE, 2018).
25. Byerly, A., Kalganova, T. & Dear, I. No routing needed between capsules. *Neurocomputing* **463**, 545–553 (2921).
26. Reuther, A. et al. Survey and benchmarking of machine learning accelerators. In *IEEE High Performance Extreme Computing Conference* 1–9 (IEEE, 2019).
27. Norrie, T. et al. The design process for google's training chips: TPUv2 and TPUv3. *IEEE Micro* **41**, 56–63 (2021).
28. Fuchs, A. & Wentzlaff, D. The accelerator wall: limits of chip specialization. In *IEEE International Symposium on High Performance Computer Architecture* 1–14 (IEEE, 2019).
29. Adolf, R., Rama, S., Reagen, B., Wei, G.-y. & Brooks, D. Fathom: reference workloads for modern deep learning methods. In *IEEE International Symposium on Workload Characterization* 1–10 (IEEE, 2016).
30. Liu, J., Zhao, H., Ogleari, M. A., Li, D. & Zhao, J. Processing-in-memory for energy-efficient neural network training: a heterogeneous approach. In *51st Annual IEEE/ACM International Symposium on Microarchitecture* 655–668 (IEEE, 2018).
31. Reddy, M. in *API Design for C++* (ed. Reddy, M.) 209–240 (Morgan Kaufmann, 2011).
32. Sarpeshkar, R. Analog versus digital: extrapolating from electronics to neurobiology. *Neural Comput.* **10**, 1601–1638 (1998).
33. Hu, M., Strachan, J. P., Li, Z. & Stanley-Williams, R. Dot-product engine as computing memory to accelerate machine learning algorithms. In *17th International Symposium on Quality Electronic Design* 374–379 (IEEE, 2016).
34. Garbin, D. et al. Variability-tolerant convolutional neural network for pattern recognition applications based on oxram synapses. In *IEEE International Electron Devices Meeting* 28.4.1–28.4.4 (IEEE, 2014).
35. Lin, P. et al. Three-dimensional memristor circuits as complex neural networks. *Nat. Electron.* **3**, 225–232 (2020).
36. Chen, J.-H., Jang, C., Xiao, S., Ishigami, M. & Fuhrer, M. S. Intrinsic and extrinsic performance limits of graphene devices on SiO_2. *Nat. Nanotechnol.* **3**, 206–209 (2008).
37. Querlioz, D., Bichler, O. & Gamrat, C. Simulation of a memristor-based spiking neural network immune to device variations. In *2011 International Joint Conference on Neural Networks* 1775–1781 (IEEE, 2011).
38. Payvand, M., Nair, M. V., Müller, L. K. & Indiveri, G. A neuromorphic systems approach to in-memory computing with non-ideal memristive devices: from mitigation to exploitation. *Faraday Discuss.* **213**, 487–510 (2019).
39. Moro, F. et al. Neuromorphic object localization using resistive memories and ultrasonic transducers. *Nat. Commun.* **13**, 3506 (2022).
40. Li, Y., Wang, Z., Midya, R., Xia, Q. & Yang, J. J. Review of memristor devices in neuromorphic computing: materials sciences and device challenges. *J. Phys. D* **51**, 503002 (2018).
41. Wang, Y. et al. Mott-transition-based RRAM. *Mater. Today* **28**, 63–80 (2019).
42. Wang, H. & Yan, X. Overview of resistive random access memory (RRAM): materials, filament mechanisms, performance optimization, and prospects. *Phys. Status Solidi Rapid Res. Lett.* **13**, 1900073 (2019).
43. Akerman, J. Toward a universal memory. *Science* **308**, 508–510 (2005).
44. Palem, K. V. Energy aware computing through probabilistic switching: a study of limits. *IEEE Trans. Comput.* **54**, 1123–1137 (2005).
45. Camsari, K. Y., Faria, R., Sutton, B. M. & Datta, S. Stochastic p-bits for invertible logic. *Phys. Rev. X* **7**, 031014 (2017).
46. Camsari, K. Y., Sutton, B. M. & Datta, S. p-bits for probabilistic spin logic. *Appl. Phys. Rev.* **6**, 011305 (2019).
47. Borders, W. A. et al. Integer factorization using stochastic magnetic tunnel junctions. *Nature* **573**, 390–393 (2019).
48. Khasanvis, S. et al. Self-similar magneto-electric nanocircuit technology for probabilistic inference engines. *IEEE Trans. Nanotechnol.* **14**, 980–991 (2015).
49. Kim, J. et al. Exploitable magnetic anisotropy of the two-dimensional magnet CrI_3. *Nano Lett.* **20**, 929–935 (2020).
50. Chen, Z., He, J., Zhou, P., Na, J. & Sun, L. Strain control of the electronic structures, magnetic states, and magnetic anisotropy of Fe doped single-layer MoS_2. *Comput. Mater. Sci.* **110**, 102–108 (2015).
51. Mizrahi, A. et al. Neural-like computing with populations of superparamagnetic basis functions. *Nat. Commun.* **9**, 1533 (2018).
52. Bhuin, S., Sweeney, J., Pagliarini, S., Biswas, A. K. & Pileggi, L. A self-calibrating sense amplifier for a true random number generator using hybrid FinFET-straintronic MTJ. In *2017 IEEE/ACM International Symposium on Nanoscale Architectures* 147–152 (IEEE, 2017).
53. Bhuin, S., Biswas, A. K. & Pileggi, L. Strained MTJs with latch-based sensing for stochastic computing. In *IEEE 17th International Conference on Nanotechnology* 1027–1030 (IEEE, 2017).
54. Pagliarini, S. N., Bhuin, S., Isgenc, M. M., Biswas, A. K. & Pileggi, L. A probabilistic synapse with strained MTJs for spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **31**, 1113–1123 (2020).
55. McDowell, D. L. et al. in *Integrated Design of Multiscale, Multifunctional Materials and Products* (eds McDowell, D. L. et al.) 351–360 (Butterworth-Heinemann, 2010).
56. Kaspar, C., Ravoo, B. J., van der Wiel, W. G., Wegner, S. V. & Pernice, W. H. P. The rise of intelligent matter. *Nature* **594**, 345–355 (2021).
57. Goswami, S. et al. Decision trees within a molecular memristor. *Nature* **597**, 51–56 (2021).
58. Zadeh, A. H., Poulos, Z. & Moshovos, A. Deep learning language modeling workloads: where time goes on graphics processors. In *IEEE International Symposium on Workload Characterization* 131–142 (IEEE, 2019).
59. Oh, S. et al. Energy-efficient Mott activation neuron for full-hardware implementation of neural networks. *Nat. Nanotechnol.* https://doi.org/10.1038/s41565-021-00874-8 (2021).
60. Surekcigil Pesch, I., Bestelink, E., de Sagazan, O., Mehonic, A. & Sporea, R. A. Multimodal transistors as ReLU activation functions in physical neural network classifiers. *Sci. Rep.* **12**, 670 (2022).
61. Mennel, L. et al. Ultrafast machine vision with 2D material neural network image sensors. *Nature* **579**, 62–66 (2020).
62. Yu, S., Jiang, H., Huang, S., Peng, X. & Lu, A. Compute-in-memory chips for deep learning: recent trends and prospects. *IEEE Circuits Syst. Mag.* **21**, 31–56 (2021).
63. Gallego, G. et al. Event-based vision: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 154–180 (2022).
64. Yao, P. et al. Fully hardware-implemented memristor convolutional neural network. *Nature* **577**, 641–646 (2020).

65. Chen, Y. et al. Polymer memristor for information storage and neuromorphic applications. *Mater. Horiz.* **1**, 489–506 (2014).
66. Salas, E. B. Number of satellites launched from 1957 to 2019. *Statista* https://www.statista.com/statistics/896699/number-of-satellites-launched-by-year/#statisticContainer (2022).
67. Tan, F. et al. Investigation on the response of TaO_x-based resistive random-access memories to heavy-ion irradiation. *IEEE Trans. Nucl. Sci.* **60**, 4520–4525 (2013).
68. Gao, L., Holbert, K. E. & Yu, S. Total ionizing dose effects of gamma-ray radiation on nbox-based selector devices for crossbar array memory. *IEEE Trans. Nucl. Sci.* **64**, 1535–1539 (2017).
69. Lupo, N., Calligaro, C., Gastaldi, R., Wenger, C. & Maloberti, F. Design of resistive non-volatile memories for rad-hard applications. In *IEEE International Symposium on Circuits and Systems* 1594–1597 (IEEE, 2016).
70. Park, G. et al. Immunologic and tissue biocompatibility of flexible/stretchable electronics and optoelectronics. *Adv. Healthc. Mater.* **3**, 515–525 (2014).
71. Salmilehto, J., Deppe, F., Di Ventra, M., Sanz, M. & Solano, E. Quantum memristors with superconducting circuits. *Sci. Rep.* **7**, 42044 (2017).
72. Spagnolo, M. et al. Experimental photonic quantum memristor. *Nat. Photon.* https://doi.org/10.1038/s41566-022-00973-5 (2022).
73. Li, X. & Wu, X. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing* 4520–4524 (IEEE, 2015).
74. Soriano, M. C. Reservoir computing speeds up. *Physics* https://doi.org/10.1103/physics.10.12 (2017).
75. Tanaka, G. et al. Recent advances in physical reservoir computing: a review. *Neural Netw.* **115**, 100 – 123 (2019).
76. Larger, L. et al. High-speed photonic reservoir computing using a time-delay-based architecture: million words per second classification. *Phys. Rev. X* **7**, 011015 (2017).
77. Fernando, C. & Sojakka, S. in *Advances in Artificial Life: ECAL 2003* Lecture Notes in Computer Science Vol. 2801 (eds Banzhaf, W. et al.) 588–597 (Springer, 2003); https://doi.org/10.1007/978-3-540-39432-7_63
78. Asanovic, K. et al. *The Landscape Of Parallel Computing Research: A View From Berkeley* Technical Report UCB/EECS-2006-183 (EECS Department, Univ. California, Berkeley, 2006); http://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html
79. Jongerius, R., Stanley-Marbell, P. & Corporaal, H. Quantifying the common computational problems in contemporary applications. In *IEEE International Symposium on Workload Characterization* 74–74 (IEEE, 2011).
80. Tsai, H., Ambrogio, S., Narayanan, P., Shelby, R. M. & Burr, G. W. Recent progress in analog memory-based accelerators for deep learning. *J. Phys. D* **51**, 283001 (2018).

Acknowledgements

P.S.-M. is supported by EPSRC grant EP/V047507/1 and by the UKRI Materials Made Smarter Research Centre (EPSRC grant EP/V061798/1). N.J.T. acknowledges funding from EPSRC grant EP/L016087/1. S.H. acknowledges funding from EPSRC (EP/P005152/1, EP/P007767/1). We thank J. Crowcroft, S. Tappertzhofen and H. Joyce for their comments and feedback on the paper. Lastly, we acknowledge the contributions of J. Meech and J. Rodowicz in compiling the data in Supplementary Fig. 5.

Author contributions

P.S.-M. conceived the idea. N.J.T. wrote the paper, collected data and performed analysis under the guidance of S.H. and P.S.-M.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41928-023-00977-1.

Correspondence should be addressed to Phillip Stanley-Marbell.

Peer review information *Nature Electronics* thanks Kerem Camsari, Sreetosh Goswami, Melika Payvand and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.