Accelerating Scientific Computing in the Post-Moore's Era

KATHLEEN E. HAMILTON, CATHERINE D. SCHUMAN, STEVEN R. YOUNG, RYAN S. BENNINK, NEENA IMAM, and TRAVIS S. HUMBLE, Computing and Computational Sciences Directorate, Oak Ridge National Laboratory, USA

Novel uses of graphical processing units for accelerated computation revolutionized the field of high-performance scientific computing by providing specialized workflows tailored to algorithmic requirements. As the era of Moore's law draws to a close, many new non–von Neumann processors are emerging as potential computational accelerators, including those based on the principles of neuromorphic computing, tensor algebra, and quantum information. While development of these new processors is continuing to mature, the potential impact on accelerated computing is anticipated to be profound. We discuss how different processing models can advance computing in key scientific paradigms: machine learning and constraint satisfaction. Significantly, each of these new processor types utilizes a fundamentally different model of computation, and this raises questions about how to best use such processors in the design and implementation of applications. While many processors are being developed with a specific domain target, the ubiquity of spin-glass models and neural networks provides an avenue for multi-functional applications. This also hints at the infrastructure needed to integrate next-generation processing units into future high-performance computing systems.

CCS Concepts: • **Theory of computation → Models of computation**; • **Hardware → Emerging technologies**; *Neural systems*; *Quantum computation*;

Additional Key Words and Phrases: Graph algorithms, machine learning, constraint satisfaction problems, quantum computing, neuromorphic computing, optical Ising machines

ACM Reference format:
Kathleen E. Hamilton, Catherine D. Schuman, Steven R. Young, Ryan S. Bennink, Neena Imam, and Travis S. Humble. 2019. Accelerating Scientific Computing in the Post-Moore's Era. *ACM Trans. Parallel Comput.* 7, 1, Article 6 (February 2020), 31 pages.
https://doi.org/10.1145/3380940

This work was supported in part by the United States Department of Defense and used resources of the Computational Research and Development Programs at Oak Ridge National Laboratory. Research sponsored in part by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U. S. Department of Energy. The work was supported in part by the Department of Energy, Office of Science, Early Career Research Program. This work was partially supported as part of the ASCR Testbed Pathfinder Program at Oak Ridge National Laboratory under FWP #ERKJ332. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under contract number DE-AC05-00OR22725.

Authors' addresses: K. E. Hamilton (corresponding author), C. D. Schuman, S. R. Young, R. S. Bennink, N. Imam, T. S. Humble, Computing and Computational Sciences Directorate, Oak Ridge National Laboratory, One Bethel Valley Road, Oak Ridge, TN, 37831; emails: {hamiltonke, schumancd, youngsr, benninkrs, imamn, humblets}@ornl.gov.

2329-4949/2019/02-ART6 $15.00
https://doi.org/10.1145/3380940

1 INTRODUCTION

Graphical processing units can be networked into clusters for distributed computing, and the use of matrix-based computations changed the landscape for scientific and high-performance computing [66, 133, 245, 261]. Most notably, the incorporation of GPUs into neural network training has allowed for significant advances in the field of deep learning and artificial learning [148, 204, 220].

With the proliferation of non–von Neumann architectures and devices, such as quantum and neural processing units, there is an expectation that these devices will lead to a similar disruption in scientific computing. These are young technologies that are not developed to be general purpose computing systems, but are designed instead to efficiently implement specific computational models. The paramount question is: On what application will these devices show significant speedup, computational advantage, or supremacy? Second, what is the domain that stands most to benefit from these new technologies? Third, how will they perform in heterogeneous workflow?

The goal of this survey is to identify several prominent computational models that are implemented by emerging processor technologies. For each identified type of processor, we summarize the class of algorithms for which they are expected to outperform current computing technologies. We also highlight secondary algorithm classes that can be deployed on the same hardware. In Section 2 to 4, we describe the hardware, computational models, and algorithms that we use to organize our discussion of four different processors. This survey focuses on four next-generation processor types: tensor processing units, Section 5; annealer-based devices, Section 6; digital quantum devices, Section 7; and neural processing units or neuromorphic systems, Section 8. For each, we discuss the computational model and focus on how each hardware can be used for two algorithm classes: constraint satisfaction problems (CSPs) and machine learning (specifically deep learning of neural networks).

This is an expanded version of a paper presented at GraML 2018 [94]. We have added a significant discussion of digital quantum devices and challenges encountered when working with next-generation processors. As these are rapidly developing fields of research, we have tried to provide a comprehensive listing of recent results and highlight other relevant survey papers when possible. For easy reference, we have included a table of acronyms defined throughout the main text in Appendix B.

2 HARDWARE DEVICES

At the hardware level, we discuss three classes of next-generation processors: quantum processing units (QPUs), neural processing units (NPUs), and tensor processing units (TPUs). NPUs and QPUs are non–von Neumann architectures. Quantum processors are devices that utilize quantum properties such as superposition and entanglement as part of their computational model and we distinguish between: annealer-based devices (ABD) and digital quantum devices (DQD). Neural processing units cover neuromorphic hardware, from semiconductor-based, to memristive, and also biological systems. We focus our discussion on the following near-term devices:

- **TPU**: Google's custom design called a Tensor Processing Unit™ [126]
- **ABD**: Annealer-based devices such as: quantum D-Wave 2X and 2000Q developed by D-Wave [58, 121], optical Ising machines, and the digital annealer developed by Fujitsu [77].
- **DQD** Digital quantum devices such as: superconducting qubit chips developed by IBM [116], Rigetti [209], or trapped ion systems developed by IonQ [55].
- **NPU**: IBM's Neurosynaptic System [168], the SpiNNaker System [78, 187], Intel's Loihi chip [154], and systems based on memristors or other emerging technologies

and include TPUs in our discussion, as they are a natural continuation of the progression from vector-based computation (CPUs) to matrix-based computation (GPUs) to tensor-based computation (TPUs).

3 COMPUTATIONAL MODELS

3.1 Artificial Neural Networks: Computing with Nonlinear Neurons

Artificial Neural Networks (ANNs) have been applied to a large variety of tasks that will be useful in a supercomputing setting. They use nonlinear neurons with weighted connections to uncover features in data, compress and retrieve data, and to generate data. Data analytics is one of the key areas in which neural networks will play a role, both in the development of analytics tools (i.e., training for neural networks) and in the deployment of existing analytics tools (i.e., inference for neural networks).

The structure of ANNs encompasses a wide variety of topologies. A general feed-forward neural network is a layered construction formed by stacking several perceptrons to build a multilevel perceptron (MLP). This is an edge-weighted, acyclic network with deterministic, nonlinear units connected with directed edges. Hidden units play a vital role in the computational power of MLPs and other ANNs. These units are not controlled by the user, and their values are determined by the weights and biases of the network. ANNs are powerful computational models and increasing the number of layers they contain (and subsequently the number of hidden units) can result in a network that can be a universal approximator if nonlinear functions [46, 109]. Currently, the most widely used discriminative model utilizes MLPs [212] or one of its derivatives, such as convolutional neural networks (CNNs) [149]. These networks are trained using stochastic gradient descent–based back-propagation. Traditional ANNs such as MLPs and CNNs map most naturally to TPU-style systems but can also be mapped to NPU systems.

Q1

Boltzmann machines are a neural network type with symmetric edges. These are ANNs composed of stochastic neural units used to define generative models. These networks are fully connected, weighted graphs and training is used to find a minimum global energy [1]. The computational complexity that results from this complete connectivity makes them impossible to train for anything but trivial problems on conventional hardware [185]. Thus, a modification called a restrictive Boltzmann machine (RBM) is typically used where the neurons are partitioned into a bipartite graph, and the network is trained using contrastive divergence [103]. RBMs can then be stacked into a deep structure to form a network called a deep belief network (DBN) [104], which is used to build generative models.

DBNs have been utilized in the literature for unsupervised feature recognition in both audio [170] and image [111, 151] data. The ability to do unsupervised feature recognition is a key capability in a variety of contexts for future supercomputing systems. For training purposes, access to a co-processor that can quickly train Boltzmann machines or restricted Boltzmann machines for unsupervised feature recognition would be extremely useful in making realistic use of those systems. This is an especially compelling use case when considering that another processor in a heterogeneous supercomputing node may be used for scientific simulations, which are generating data, and a co-processor that could perform training and inference with a Boltzmann machine on that data could operate in parallel, resulting in Boltzmann machines trained on data as it is generated, or Boltzmann machines performing feature recognition on data as it is generated. RBMs and DBNs map most naturally to systems that can realize stochasticity. In this case, they map most naturally to QPUs.

Recurrent neural networks retain the general layered construction of a feed-forward network, but introduce connections within layers. These networks, such as long short term memory

networks (LSTMs) [105], are able to model temporal relations in data. Recurrent neural networks have seen much success in providing a new state-of-the-art for domains ranging from computer vision [98] to speech processing [10]. Like traditional ANNs, recurrent neural networks and LSTMs map most naturally to TPUs.

The Hopfield network is a class of recurrent neural network with completely symmetric connections and no hidden units [106, 107]. An edge-weighted network can store a set of binary or bipolar patterns that is retrieved by supplying a candidate pattern that is then completed to match a stored pattern. We choose to include it in this section based on how it is trained. These networks are distinguished from other ANNs in that the weights associated with a model are determined through various "learning" rules, not through training over data. There are a number of learning rules for the synaptic weights, including: Hebb rule learning [106], Willshaw learning [82], Storkey learning [247], or projection learning [150].

Hopfield networks can be utilized as an auto-associative memory model known as a content-addressable memory (CAM) model [106]. The edge weights of the network are determined using Hebbian learning (or another learning rule), fixing the units contained in the key, while the network updates the remaining units to complete the pattern. For this application, Hopfield networks can be considered both as CSP or ANN: They are constructed using nonlinear neurons connected by weighted edges, and pattern matching is implemented by finding a pattern that minimizes the network energy functional.

The topology of the network for a CAM model of order N is a fully connected clique K_N, and the storage capacity with perfect recall is upper bounded by $0.15N$ [9, 106, 108, 166]. A bidirectional memory model stores patterns of order N in the edge weights of a fully connected biclique $K_{N/2,N/2}$ [140]. Multiple learning rules [82, 150, 247] and network topologies [7, 8, 22, 86, 129] have been developed with the goal of increasing the pattern storage capacity of associative memory models.

3.2 Spiking Neural Networks: Event-driven Computing

Spiking neural networks (SNNs) are a third type of artificial neural network that incorporate time into the way that the network processes information [159], most notably by computing through the transmission and reception of electrical pulses ("spikes"). In SNN systems, the times at which inputs are received in the system matter in how the network processes that information. Neurons in SNNs may be deterministic or stochastic, giving a degree of flexibility in the way computation is done in these systems.

One of the use cases of SNNs is to emulate biological neurons to execute brain-like computing, as they are more closely aligned with how biological brains operate than traditional artificial neural networks. The need for more brain-like characteristics has driven device development in terms of supporting synaptic plasticity [51] and having neuron models that can emulate neuronal dynamics [38]. Spiking neural networks can be constructed and trained like ANNs by modifying training methods such as back-propagation to allow for spiking data, and this has been the dominant application for NPUs. However, they have also been trained by a variety of other mechanisms, including unsupervised, Hebbian-style learning [246] and evolutionary optimization [229].

As a computational model, spiking recurrent neural networks with synaptic plasticity mechanisms—which are learning mechanisms that update the synaptic weights based on activity in the network—have been shown to have super-Turing capabilities [34, 241]. More precisely, these types of networks are capable of solving a class of problems beyond those that can be described with Turing machines. Though they are more powerful in theory, in practice, the algorithms required to assemble, train, or learn these types of networks have not yet been fully developed or

investigated by the community, which has restricted their widespread use. NPUs are the natural platform for SNNs, as most NPUs directly implement in hardware some form of an SNN.

3.3 Ising Model: Computing with Interacting Spins

In the Ising or "spin glass" model, information is stored in pairwise correlations between interacting variables called spins. Spin glasses are most commonly used to model complex probability distributions or, via controlled evolution of the interaction parameters, to solve minimization problems. The latter process is generally called annealing in analogy to the physical process of relaxing a material into its energetically optimal configuration by slowly lowering its temperature. Solving an optimization problem with a spin glass first requires the objective function to be expressed as an Ising (or Potts) energy function called the Hamiltonian. The goal is then to find the configuration of spins that minimizes the Hamiltonian. This configuration, called the ground state, is determined by the pattern of positive and negative couplings between spins.

Graphically, spin glass models are undirected graphs with weighted edges and vertices. They can be constructed with binary or bipolar spin states (Ising) or with multilevel spin states (Potts). The theoretical development of Ising-glass and Potts-glass models has grown from the study of spin dynamics in magnetic and paramagnetic materials. These materials can have novel dynamics due to the competition between local and global spin-spin interactions, often exhibiting sharp transitions between ordered and disordered phases [237]. Spin glasses can be used to solve not only optimization problems, but constraint satisfaction problems and a variety of other NP-hard problems [101, 158]. The transition between ordered and disordered phases generally corresponds to a transition between "easy" and "difficult" regimes of a computational problem [110]. Spin glass systems and SNNs share a common feature of parallelization, where a single unit is able to affect change in multiple neighboring units simultaneously. Spin glasses are easily mapped to ABDs.

3.4 Quantum Circuits: Computing with Superpositions of Logical Values

Quantum computing promises new capabilities for processing information and performing computationally hard tasks. This includes significant algorithmic advances for solving hard problems in computing [172], sensing [56], and communication [142]. The breakthrough examples of Shor's algorithm for factoring numbers and Grover's algorithms for unstructured search have fueled a series of more recent advances in computational chemistry, nuclear physics, and optimization research among many others. However, realizing the algorithmic advantages of quantum computing requires hardware devices capable of encoding quantum information, performing quantum logic, and carrying out sequences of complex calculations based on quantum mechanics [183].

The quantum circuit model is the oldest and most-studied quantum computational model. Analogous to a conventional digital circuit, a quantum circuit is a sequence of primitive logical operations on a set of quantum mechanical bits, whose net effect is to transform an input value into an output value. But in contrast to bits, quantum bits ("qubits") have the peculiar ability to exist in multiple logical states at the same time. More precisely, a set of qubits can encode any linear combination of their possible logical values. Such a *superposition state* is in many ways like a probability distribution, but it can be manipulated in more complex ways. The upshot is that a set of qubits has an exponentially larger state space and supports a richer set of logical operations than the same number of bits. Consequently, quantum circuits are believed to be much more powerful than classical (non-quantum) computers for certain computational tasks. An important example of such a task is finding hidden subgroups, which can be accomplished efficiently by a quantum circuit to yield an efficient factoring algorithm [239]. Quantum circuits are also expected to significantly outperform classical computers at finding preimages of functions, sometimes described

as unstructured database search [87]. The circuit-based model of quantum computation is directly implemented on DQDs.

4 ALGORITHM CLASSES

The storage and retrieval of information in next-generation processors makes it difficult to define matrix-based data structures and computation that are utilized in most CPU-based algorithms. For quantum ABDs and NPUs (without synaptic plasticity) the information cannot be updated in-place. Quantum ABDs stores information in the weights of couplers and spin states; NPUs rely on discrete spikes and can only store information in synaptic weights. Each of the models in Section 3 have characteristics that make them well-suited to different types of algorithmic tasks, but they can all be implemented or simulated on CPUs and GPUs. A significant amount of research is aimed toward identifying algorithmic tasks that will benefit from next-generation processor hardware.

For example, the theoretical advantages of quantum algorithms over classical algorithms have been derived for many tasks, including integration, matrix inversion, finding eigenvalues and eigenvectors, constraint satisfaction, combinatorial optimization, and sampling.[1] The wide variety of applications that could benefit from accelerating such core computational tasks makes the quantum circuit model of computing very appealing. However, this model requires either extremely well-controlled qubits or a very large number of imperfectly controlled qubits in conjunction with a sophisticated scheme for fault-tolerant computation. At the time of writing this survey, we are not aware of any devices capable of supporting either approach.

We focus on two applications, constraint satisfaction problems (CSPs) and machine learning (ML), which have incorporated next-generation processors in distinct ways. For CSPs, a model is constructed off-line (commonly referred to as pre-processing) and then implemented on a next-generation processor to speed up time to solution. For machine learning, next-generation processors can be used to speed up the execution of a trained ANN; they can be used to make the training of an ANN more efficient; or the training process may use the next-generation processor in the loop to significantly speed up model development. In Section 5 to 8, we will discuss each of the next-generation processors in terms of the computational model and algorithmic class, and also identify potentially new use cases.

4.1 Constraint Satisfaction Problems

CSPs represent an important class of hard problems for scientific computing, in which a satisfying variable assignment maximizes a defined objective while adhering to specified constraints. In general, the worst-case problem complexity is categorized as NP-Complete, but many problem subclasses can be solved in polynomial time. In addition, approximate solutions can be recovered that either do not satisfy all the necessary constraints or do not find the global maximum value. The approximation ratio measures the ratio of the observed value to the actual maximum, and efficient methods that provide guarantees on the expected approximation ratio are useful for bounding practical results.

CSPs such as graph coloring and 3-SAT belong to the computational complexity class NP [160, 173, 219]. A wider range of problems, including traveling salesman, are classified as NP-hard. Both NP and NP-hard problems cannot be solved through reduction to a simpler problem, but instead require an exhaustive search over a large set of possible solutions (which may be exponentially large). Many algorithms have been developed to solve CSPs [145], and in particular, simulated annealing is a well-established algorithm for solving constraint satisfaction problems that has been realized in software using conventional computing platforms. Simulated annealing [138, 255] can

[1]We direct the reader to the Quantum Algorithm Zoo for an exhaustive and up-to-date listing of quantum algorithms [123].

Table 1. Neural Network Characteristics, Applications, and Target Hardware

Computational Model	Hidden Units	Network Characteristics	Training Method	Hardware
Spin-glasses	No	Recurrent, deterministic	Heuristic (classical)	QPU-ABD, NPU
RBMs, DBNs	Yes	Recurrent, stochastic	Contrastive divergence (classical)	QPU-ABD, NPU
SNNs	Yes	Recurrent, temporal dynamics stochastic or deterministic	STDP (spiking), gradient descent (classical), evolutionary algorithms (classical)	NPU
ANNs	Yes	Deterministic	Gradient descent (classical), Back-propagation (classical, spiking or quantum)	GPUs, TPUs, NPUs
QCL	NA	Stochastic(due to device noise)	Gradient descent (quantum-classical), adversarial (classical or quantum) Gradient-free methods (quantum-classical)	QPU-GBD

be used to solve optimization problems, utilizing the phase transitions in a spin-glass system to find a global optima.

On a conventional processor the search for the ground state can be executed through various simulated annealing methods or Markov Chain Monte Carlo methods. The fundamental challenge faced when solving these problems on conventional processors is the possibility that the search terminates at a locally optimal configuration that is substantially different than, and inferior to, the global optimum. It is hoped that many next-generation processors can offer significant computational advantages in solving these problems.

4.2 Machine Learning and Deep Learning

ML refers broadly to a set of algorithms or techniques that "learn" from data, rather than being programmed. Deep Learning is a diverse field of research and covers the algorithms used to train the ANNs described in Section 3. These learning methods can be supervised, semi-supervised, or unsupervised [148]. There are also a number of biologically inspired models such as evolutionary learning or reinforcement learning. Examples of these techniques include decision trees, k-nearest neighbors, and support vector machines.

Training an ANN determines the weights and biases in the set $\theta = \{w, b\}$. For large ANNs this is a computationally intensive task that has been significantly accelerated through the use of GPUs. Large networks are generally trained using gradient descent (or stochastic gradient descent), where the parameters are iteratively updated depending on the gradient (or an estimate of the gradient) of the cost function. Generative models (such as RBMs) are trained with gradient-based estimation methods such as Gibbs sampling or contrastive divergence. Training ANNs is also an optimization problem, as the search focuses on identifying the parameter set that minimizes (or maximizes) a given cost function $J(\theta)$.

Table 1 gives a brief summary of the differences of the computational models that are variants of ANNs, how they are trained, and the most appropriate post Moore's era technologies for those network types. Training that is done on a CPU or GPU is characterized as "classical" and can be done prior to deployment on another processor, or can be incorporated into a hybrid training algorithm. Training that is implemented solely on a next-generation processor is distinguished by the computational model (i.e., spiking or quantum).

5 TENSOR PROCESSING UNITS

Currently, the most computationally intensive and widely used types of neural networks in deep learning research are CNNs and LSTM networks. Like their predecessor, the MLP, the overwhelming majority of the computation used for inference lies within a weighted-sum computation that can be mapped to multiply and accumulate operations. Thus, for a wide variety of deep learning approaches that are rapidly being deployed to production systems, there exists a shared computational unit at their core. With the surge of use of these networks, a custom ASIC implementation of this computation is now becoming an economically feasible proposition for many large data companies. As such, researchers at Google have developed a custom design called a Tensor Processing Unit^{TM}(TPU) [125, 126].

The current generation TPU focuses heavily on concerns that could previously be ignored when deep learning computation was largely done as research or batch process runs against data periodically. Stakeholders are now looking to utilize deep learning in near real time. They are looking to collect data (e.g., speak into their phone), have the deep learning system operate on that data (e.g., transcribe voice to text), and then be provided actionable information on that data (e.g., a list of relevant search results). Such a system demands low-latency, low-power consumption, and high-throughput. To provide a low-latency inference system, the TPU is designed with a deterministic execution model, rather than a model that incorporates optimizations such as cache or out-of-order execution, which can come at the expense of guaranteed latency. This design allows the TPU to meet latency requirements while achieving throughput improvements of over 15×–30× and a power efficiency improvement of 30×–70× over current CPUs and Kepler generation GPUs [126]. Moreover, the TPU designers have proposed some minor changes that would allow future designs to achieve 3× improvements in throughput while more than doubling the power efficiency. While there may be concerns about using the reduced precision computation available in TPUs, there has been work in utilizing the lower-precision computation to perform higher-precision computation through methods such as iterative refinement [100]. This work aims to bring the accelerated computation offered by low-precision hardware to scientific applications requiring higher precision than commercial machine learning applications.

This system represents the current state-of-the-art for deep learning hardware. However, it has been noted that many NP hard problems are tensor in nature [102] and recently, CSPs have been constructed for execution on a TPU [14].

6 ANNEALING DEVICES

Simulated annealing is a common method used to solve CSPs. Annealer-based devices have been used to accelerate the solving of CSPs by relying on computational models that can natively implement annealing through low-level dynamics of the hardware.

6.0.1 Quantum Annealers. Currently, there is a family of quantum annealers from D-Wave Systems [58, 121]. In this design family, which contains the 2X and 2000Q devices, the interacting qubits are connected in a fixed geometry known as a Chimera structure (see Figure 1) that can be conceptualized as a graph with weighted vertices and edges. The current D-Wave 2000Q processor contains approximately 2,048 qubits. The sparsity of the fixed Chimera hardware graph limits the size of problems that can be solved with the D-Wave system [139], though many approaches to the "embedding problem" have been developed [24, 35, 41, 42, 90, 265].

The D-Wave system can be used by researchers to explore the physics of Ising spin systems. Recent studies have been carried out that demonstrate critical behavior of quantum systems with over 400 simulated qubits [135] as well as three dimensions [96]. Other quantum annealing platforms have been developed, mainly for the simulation of quantum systems [19] of up to 51 qubits.

6.0.2 Optical Ising Machines. Optical Ising machines (OIMs), also called coherent Ising machines [161], are nascent computing devices that use a process similar to annealing to find the minimum energy state of a spin glass. Each binary variable or spin is encoded in the phase (0 or π) of an optical field. Starting from quantum fluctuations, the fields are amplified within an optical resonator while being mixed according to the coefficients of the spin coupling matrix. This causes field configurations that correspond to low-energy spin states to grow in amplitude at the expense of field configurations that correspond to high-energy spin states [263]. After a few hundred cycles around the resonator, the fields converge to a steady configuration. Measuring the phases of the fields then yields a spin configuration that with high probability has a low Ising energy.

OIMs have been demonstrated to solve MAX-CUT problems (a type of combinatorial optimization problem) and to sample from Boltzmann distributions with performance comparable to that of state-of-the-art quantum annealers [88, 118, 167, 251]. While it was initially conjectured that OIMs may be a type of quantum computer [95, 238, 250], some recent studies suggest that OIMs may be best understood as a type of gradient-descent optimizer [43, 134].

In contrast to many of the near-term processors considered in this article, OIMs are based on mature technology and can be constructed largely from commercial off-the-shelf components. Another attractive feature of OIMs is that they do not require highly specialized facilities or operating conditions: mechanical stability and optical isolation are all that is required. This eases the incorporation of OIMs into HPC environments.

6.0.3 Digital Annealers. Recently, Fujitsu has announced the development of a hardware-based annealer implemented initially using a conventional FPGA and later with CMOS technology. Unlike the implementations of quantum annealing, the Fujitsu device is reported to support all-to-all connectivity between 1,024 binary register elements. Therefore, the technology should be capable of operating at room temperature. Details about the technology are currently limited [11, 25, 77].

6.1 Application to CSPs

Spin glass models have found success in a number of algorithms, but the search over all spin states is not guaranteed to find the global energy minimum. Simulated annealing relies on thermal vibrations to search over the energy landscape, but these may not be sufficient to allow the system to escape spurious minima. Similar to simulated annealing, quantum annealing finds the solution of an Ising system by searching over its energy landscape [4, 68, 70] and can be applied to many NP hard problems and, specifically, quadratic unconstrained binary optimization problems [181, 208].

Quantum annealing is expected to have significant computational advantages over simulated annealing by leveraging quantum effects such as tunneling through barriers and superposition to escape local minima [57, 137]. QPU-ABDs have been used to solve SAT problems and CSPs [60, 136, 216] and other general optimization problems [40, 99, 120, 256]. Programming the D-Wave ABD to solve an unconstrained optimization problem requires first constructing a Hamiltonian for an Ising spin glass system, which defines the needed edge weights (couplings) and vertex weights (biases) [112, 256]. The processor then operates by evolving a well-known initial quantum state to the corresponding ground state of the programmed Ising Hamiltonian [158]. Measurement of the prepared state provides a candidate solution for the programmed optimization problem.

However, the full capability of quantum annealing and identifying problems that could arise as systems are scaled up still remain an active area of debate [5, 6, 13, 131, 146, 193].

6.2 Application to ML

The most computationally intensive part of the contrastive divergence algorithm for unrestricted Boltzmann machines is the annealing to reach the thermal equilibrium, followed by sampling at

the equilibrium state to estimate the probabilities required for contrastive divergence. The D-Wave quantum annealing processor performs this type of calculation naturally. Due to the Chimera topology in the D-Wave, however, a fully connected Boltzmann machine utilizing all qubits is not realizable; to get around this issue, Potok et al. proposed a limited Boltzmann machine, which has greater connectivity than the restricted Boltzmann but still operates within the constraints of the D-Wave configuration [198]. Utilizing the D-Wave as a co-processor allows training for the limited Boltzmann machine that would take months on a conventional von Neumann architecture to occur in minutes to hours. Because Boltzmann machines have more sophisticated representations due to their greater interconnections, their data analysis capabilities are theoretically greater as well. QPUs will allow researchers to apply Boltzmann machines to real tasks in a reasonable amount of time.

The quantum Helmholtz machine introduced in Reference [17] is an example of utilizing quantum annealers as part of a hybrid classical-quantum workflow. In this case, a DBN is constructed with most layers being deployed on classical processors, and only the final layer deployed on a quantum annealer. This approach is able to avoid several of the challenges described in Section 2 such as limited connectivity and bits of precision.

Classical associative memory models models can be mapped to quantum systems for pattern retrieval [231]; the use of quantum annealing simplifies the search for a stored pattern by replacing the iterative update rule with a single system annealing [9, 82, 140, 166]. While quantum annealing is projected to significantly increase the capacity of an associative memory model [217], finding sparser representation of the Hopfield network can also improve to associative memory models for quantum annealing. For bidirectional memory models on quantum annealers [222] the choice of learning rule affects the performance of the model, as well as allows for flexibility in the face of limited bits of precision.

7 DIGITAL QUANTUM DEVICES

The computational model underlying DQDs differs from TPUs, NPUs, and ABDs, as it is a general purpose model of computation that enables a wide diversity of algorithms. In particular, the underlying quantum circuit model affords a digital representation of computation, in which individual qubits are manipulated through the application of a discrete sequence of unitary gates. A number of algorithms developed within the quantum circuit model have been shown to scale better than best-in-class algorithms for conventional circuit approach. The prominent examples of Grover's unstructured search algorithm and Shor's integer factoring algorithm emphasize that quantum computing has the potential to vastly outperform classical computers. However, currently available devices have been limited by the intrinsic noise and relative small size of the quantum processor. Efforts to realize these algorithms are expected to require hardware that supports fault-tolerant operations, which has a strong theoretical development but has yet to be demonstrated experimentally. In addition, the cross-over point in problem size for which a quantum algorithm finds the solution faster than a conventional approach has yet to be determined tightly. Current estimates suggest fault-tolerant quantum computers will require many orders of magnitude improvements in both the capacity and fidelity of the quantum register. However, there have been several recent studies that claim that a quantum speedup or advantage can be observed with small or shallow circuits for machine learning tasks [210] or solving linear algebra problems [26].

Consequently, current efforts to demonstrate the principles of quantum computing are focused on so-called noisy, intermediate-scale quantum (NISQ) devices [199], which loosely define a category encompassing the foreseeable future of faulty operations. NISQ devices lack the capacity and accuracy necessary to sustain to fault-tolerant operations, but they are a necessary step in such

developments. Many different technologies have been developed to implement physical qubits, such as trapped ion systems [31] and superconducting qubits [141].

For more than 35 years, there has been a broad array of experimental efforts to build quantum computing devices to demonstrate these new ideas. Multiple state-of-the-art engineering efforts have now fabricated functioning quantum processing units (QPUs) capable of carrying out small-scale demonstrations of quantum computing. Prominent examples of commercial NISQ devices include the family of cryogenic superconducting electronics processors developed independently by a number of laboratories including IBM [116], Rigetti [209], Alibaba [267], and Google [132], which are based on variants of the transmon qubit design. Other examples includes the trapped ion processors being developed by IonQ [55] or the photonic processors developed by the companies Xanadu [215] or PsiQuantum [201]. These QPUs are among a growing list of devices that have demonstrated the fundamental elements required for quantum computing [155]. This progress in prototype QPUs has opened up new discussions about how to best utilize these nascent devices [27, 29].

Current NISQ devices have on the order of 5–50 register elements (for examples, see Figure 2), gate operations with error rates from 0.1% to 0.01%, and decoherence times that are $\tilde{5}$00-2000× the gate duration. Noise in NISQ devices comes from unwanted couplings between the registers, which can interact uncontrollably with the surrounding material as well as other registers. In addition, errors in the basic operations arise from fluctuations in the applied fields as well as unintended induced interactions with the environment. These effects collectively limit the scale at which such registers and operations can perform, though efforts are underway to mitigate or reduce this noise.

Despite the performance limitations of NISQ devices, it remains possible to use them for demonstrations of small-scale applications. Modeling and simulation of quantum mechanical systems has played an important role in presenting a variety of problems that map easily into quantum computing paradigm, largely because they share the same underlying physical principles. For example, several groups have validated the use of NISQ processors for calculating the electronic structure of small molecules within standard chemical accuracy. Although these demonstrations are far from surpassing conventional methods, the ability to recover the correct answer from NISQ devices represents a milestone in the development toward solving more complex systems. Additional examples arise in modeling and simulation of nuclear and high-energy physics, where accurate modeling of quantum mechanical interactions between fundamental particles is essential to solution accuracy.

Underlying the methodologies for modeling and simulation on NISQ devices is a relatively straightforward mapping of the quantum mechanical data structure, i.e., the Hamiltonian that describes the interactions between physical particles such as electrons and nuclei, into the quantum circuit model. For static properties, such as electronic or nucleon structure, this reduction is effectively cast as an optimization problem for finding the quantum state representing the lowest-energy eigenstate of the model Hamiltonian. The variational quantum eigensolver (VQE) approach relies on the guarantee offered by the variational principles from quantum mechanics, which assert that only the true lowest-energy eigenstate, i.e., the ground state, can minimize the energy of a model Hamiltonian. Using VQE with NISQ devices becomes an exercise in accurately preparing quantum states with the quantum computer, and several examples have demonstrated this is possible by leveraging the judicious choice of so-called ansatz states that can be parametrically tuned [127, 165, 186]. Recently, studies have shown how the quantum approximate optimizer algorithm (QAOA) can also be applied to the simulation of quantum dynamics [257].

Minimizing the observed energy requires a search through the classical parameter space guided by the experimentally observed energy. In the absence of prior knowledge about the solution, this search becomes a costly bottleneck for even small molecules modeled with modest basis sets.

7.1 Applications to ML: Quantum Machine Learning

In this section, we cover recent advances in quantum machine learning as it has been developed for DQDs—specifically the circuit model of quantum computing. This area of research has been developed along multiple paths: casting elements of classical machine learning into quantum circuits, developing quantum circuits that can emulate classical machine learning algorithms, and developing standalone quantum algorithms that can speed up classical machine learning. Any advantage from quantum computing is expected to come from how the Hilbert spaces of quantum circuits and the quantum mechanical property of superposition are poised to accelerate known (classical) machine learning methods through increased representational power and parallelization. For example, there are a number of quantum algorithms that utilize superposition and quantum Fourier transforms to significantly reduce the computational overhead associated with gradient calculation [124, 258]. Full utilization of these properties requires fault-tolerant hardware or more qubits are then available on near-term NISQ devices. We refer the reader to the extensive survey papers that have been published [20, 61, 192, 226] for a fuller picture of the QML landscape.

Section 3 defined ANNs as a computational model built upon nonlinear neurons and perceptrons. Recent studies have shown how classical perceptrons [72, 225, 227] and ANNs can be implemented as quantum circuits and deployed on near-term devices [249]. Many quantum algorithms that emulate classical ML algorithms, such as reservoir computing and echo state machines [76, 80, 169], or quantum implementations of back-propagation [258, 259]. Alternatively, the studies we highlight below are focused on algorithms developed for or deployed on NISQ devices and utilize "data driven circuit learning" [15] in which a parameterized quantum circuit is not constructed from an explicitly defined Hamiltonian. These methods follow closely from the VQE methods used in simulation and the trained quantum circuits can be incorporated into generative or discriminative models. We distinguish these applications from general quantum state preparation or quantum system simulation (e.g., quantum chemistry applications), because the parameterization of the quantum circuit may not rely on a known Hamiltonian.

Generative modeling with data-driven circuit learning is effectively training a quantum circuit to prepare a target probability distribution. In this task, the exponentially large Hilbert space can be spanned by a few qubits, and the ability to simulate and sample from distributions may be difficult to compute classically. This has led to the development of quantum generative adversarial networks [49, 157] and quantum circuit Born machines [15, 156]. Many recent studies of implementing generative models on NISQ hardware [15, 16, 89, 156, 242, 266] have utilized the hardware-efficient ansatz [127]. The layers of rotation gates and entangling gates encode a high degree of nonlinearity into the circuit.

A significant task in machine learning is classification and the training of discriminator models. With near-term NISQ devices there has been a focus on how to construct quantum circuits that can classify classical data. In Reference [205], the proposed quantum advantages are twofold: A kernel classifier is expected to exhibit a quantum speedup in the runtime due to the use of amplitude amplification routines, and the exponential size of a quantum Hilbert space means that nonlinear kernels can be constructed with few qubits. References [72, 224] considered quantum classifiers that could be feasibly run on near-term hardware by restricting the number of qubits (<20), the choice of gates (one- or two-qubit unitary operations) and the overall circuit size. Reference [224] studied how the model size (number of trainable parameters) scaled with respect to the number of qubits in the circuit for a number of binary and multi-class classification tasks and also incorporated gradient-based training of circuits that could be deployed on near-term hardware. In Reference [72], a classifier was constructed as a 17-qubit circuit and used to classify the MNIST dataset (downsampled into 16 pixel images). In addition, Reference [72] introduced the concept of "quantum batch training" in which the classical data for training is put into superposition states.

The studies in References [72, 224] rely on numerical simulation of quantum circuits, but there are several recent studies of quantum classifiers that use circuits deployed on NISQ hardware using well-known benchmark datasets [84] or synthetic data [97]. In Reference [84], a circuit with 4 qubits was constructed using the 4 features of the Iris dataset [73]. This classifier was trained classically, then deployed on the IBM ibmqx4 5-qubit chip. For a binary classification task (distinguishing between 2 of the 3 iris species) the model was able to reach an accuracy of 96.5% even in the presence of device noise. In Reference [97] a 2-qubit binary classifier was trained on synthetic data and deployed on IBM hardware. This study considered two methods to incorporate quantum circuits into a classifier model. The first approach trained a parameterized quantum circuit on hardware via supervised learning and then the class label is assigned by a applying a threshold on the final sampled distribution. The second approach used a quantum circuit to construct the kernel function of a larger SVM. For both approaches, the models were able to reach near 100% accuracy on synthetic datasets of 20 points. The lowest accuracy of the second method reported as 94.75%.

7.2 Application to CSPs: Quantum Approximate Optimization Algorithms

QAOA is another candidate algorithm for NISQ devices. It is an approach for finding the minimizing assignments for a well-defined objective function. This approach casts the objective functions into a Hamiltonian formalism such that the lowest-energy quantum states is again the satisfying assignment. QAOA shares similarities with VQE [127, 165, 195], as a classical parameter search is necessary to identify the quantum state that optimizes the defined objective. Therefore, the overall computational complexity depends on both the conventional and quantum processing steps as well as the approximation parameter.

QAOA introduces an approximation parameter that can be used to control solution quality by discretizing the search over the manifold of quantum states. In particular, QAOA offers a guarantee for the approximation ratio with respect to a controllable approximation parameter p. As p approaches infinity, the approximation ratio approaches unity. However, increasing p corresponds to an increase in the complexity (depth) of the digital circuit. This approach has been applied to the study of synthetic 3-SAT problems as well as related graph problems. QAOA can be used for approximately solving CSPs by using a quantum superposition state in the search for potential assignments [67, 71]. Although the method depends strongly on the search step, there is some evidence that efficient heuristics may be possible for certain problem classes.

QAOA for small value of the approximation parameter p can be tested using very small CSP examples on NISQ devices [69, 264]. While the necessary qubits and gates are relatively small for these examples, the implementation is limited by the noisy operations supported by the NISQ devices. Moll et al. have recently evaluated QAOA to solve the CSP for MAX-Cut on a simple graph over four nodes [171]. This demonstration used the IBM quantum circuit simulator to verify that the correct solution is found with greater than 95% accuracy.

8 NEURAL PROCESSING UNITS

The dominant form of NPUs in the literature are spiking neuromorphic systems [230]. The primary purpose for utilizing an NPU is to provide either accelerated simulation of spiking neural networks or provide very lower power execution (or both). The current generation of neuromorphic hardware that is relatively widely available are digital systems such as IBM's TrueNorth Neurosynaptic Processor [168], SpiNNaker [53], and Intel's Loihi [51]. All three of these systems can implement many known classes of neural networks used in machine learning, such as CNNs, RBMs, and liquid state machines [50, 53, 63, 65, 191]. Programming of neuromorphic hardware requires the specification of many parameters required to implement systems of coupled spiking neurons.

Physical hardware custom chip implementations typically have more restrictions on connectivity, which can make programming more difficult. For example, on the TrueNorth chip, the layout and available connections show dense local connectivity (intra-core) and sparse global connectivity (inter-core). In the current hardware there are limited bits of precision allocated for specifying the various system parameters. We expect that other neuromorphic hardware systems (as they are made available) will be similarly highly optimized for a particular network model and topology and have their own sets of restrictions. Because of these potential restrictions in implementation, there are non-trivial issues associated with using conventional neural network training algorithms to produce neural networks for use on neuromorphic hardware platforms.

Though TrueNorth, SpiNNaker, and Loihi are fully digital systems, there is a large variety of spiking neuromorphic hardware implementations that utilize a variety of novel emerging device types to implement neuromorphic systems, including metal oxide memristors [39, 200], superconducting optoelectronic circuits [235], and biomolecular memristors [176]. Systems that include memristors and other emerging device types typically consume much less energy than their fully digital counterparts, which are already more energy-efficient than other, more traditional architectures. Since energy and power consumption will likely be limiting factors in post Moore's era supercomputing, novel neuromorphic systems that consume significantly less energy than most computing systems will be well-suited to a future supercomputing environment.

It is worth noting that one of the motivating factors for developing neuromorphic systems is to perform simulations of biological neural systems, which, for example, has motivated the neuromorphic system development in the Human Brain Project [36]. Large-scale, efficient, and real-time or near-real-time neuroscience simulations are still one the overarching goals for neuromorphic systems, though it is clear that there is still significant work to be done to achieve this goal [254].

8.1 Application to Deep and Machine Learning

Spiking neural networks can be constructed in the same manner as ANNs. However, executing deep learning with an SNN deployed on neuromorphic architecture requires modifying the network construction as well as the learning algorithms to incorporate spike-based signals. SNNs typically implement a learning rule called Spike Timing Dependent Plasticity (STDP), which adjusts synaptic weights in the network based on the firing activity in the network [246]. However, SNNs can also be trained using a variety of other mechanisms, including gradient descent–based methods [23, 233, 240], which adapt the training method in some way to accommodate SNNs, and evolutionary algorithms [190, 229]. Spike-based implementations of back-propagation [64, 178], CNNs [65], Gibbs sampling [50], RBMs [191], DBNs [248], and recurrent neural networks [59] have been developed for IBM's TrueNorth Neurosynaptic System and the SpiNNaker system.

An increasingly common use case of spiking neural networks on neuromorphic systems is as the "liquid" in a liquid state machine [143, 144]. Liquid state machines are a special use case of a recurrent spiking neural network with a read-out layer that is similar to the final layer in a traditional neural network [177]. Liquid state machines have been successfully used to analyze spatio-temporal data such as speech [221] and video [33]. The "liquid" itself is an untrained recurrent spiking neural network; the read-out layer is typically trained using some form of gradient descent or back-propagation. A co-processor that implements a liquid state machine could perform inference on spatio-temporal data as the data is being generated by another processor in the environment. Unlike many of the other discussed models, liquid state machines are very well-suited to deal with temporal data and could be used in combination with other neural network models for data analysis. Because the recurrent spiking neural network is untrained, it can be deployed immediately onto any spiking neuromorphic system that can realize sufficiently complex networks.

8.2 Application to CSPs

While NPUs can efficiently execute neural networks, the training of these networks is usually done on conventional processors or GPUs. There has been growing interest in applications for NPUs that avoid the need for pre-training of large neural networks. This has led to the mapping of CSPs and label propagation algorithms onto NPUs. The main challenge in adapting difficult computational problems onto NPUs is how to translate spiking data into relevant information.

CSPs have been mapped onto the SpiNNaker system and solved using systems of stochastic neurons to minimize an energy function [74, 122]. Whereas QPUs utilize quantum effects to escape spurious minima, the spike-based annealing used in References [74, 122] traverse energy barriers using intermediary network states. Pattern recall tasks have been mapped to spiking neuron systems [2, 45, 130, 159, 244] and deployed on many different neuromorphic systems [12, 194, 232].

9 INTEGRATION WITH HIGH-PERFORMANCE COMPUTING

These emerging devices are still very early in their development as computational engines, and there is an immediate focus to developed tools and libraries that enable users to interface directly with the underlying hardware. This provides expert users the ability to construct optimal programming patterns and instruction follows that may be later abstracted. However, even within this expert community, there is a need to establish a common set of tools that improve user efficiency, program reproducibility, and performance comparisons. Several Pythonic modules and libraries have been made available recently to support testing and evaluation of quantum machine learning tasks on gate-based hardware [18] or targeted for specific hardware (e.g., IBM's Qiskit Aqua library [117], Rigetti's PyQuil library [243], D-Wave [188]) as well as quantum hardware-agnostic libraries [83, 163]. Similar libraries have been developed for neuromorphic hardware control and device interfaces, including PyNN [52], Nengo [179], the TENNLab software framework [196], and Fugu [3].

Users building applications that take advantage of these devices will depend on the available interfaces. However, current library approaches are limited to relatively narrow use cases for interfacing with either the QPUs or NPUs based on currently identified workflows. Extending existing software packages presents a non-trivial burden on the users, as low-level, hardware-centric interfaces are presently the norm for both QPU and NPU programming. Widespread adoption of these new types of devices is anticipated to require a more robust ecosystem of software tools and languages that enable broader device programmability and usability.

The future integration of these next-generation processors into high-performance computing systems will require much tighter coupling between communication and memory subsystem than is currently available [260]. These interfaces must not only be compatible with devices operating under potentially disjoint computational models, but they must account for the operational constraints and programming models [114]. As a case in point, modern examples of CPU-GPU computing systems are facilitated by many different methods for enabling multi-platform communication and control, including application programming interfaces such as OpenMP, OpenCL, CUDA, and OpenACC. For the emerging devices considered above, there is a lack of best practice in the field of programming and execution that would be needed to guide similar communication and control libraries [162]. TPUs are the closest to being fully integrated into HPC workflows, while recent work on developing CPU-QPU hybrid computing systems underscores the difficulty facing those approaches [18, 28, 30, 163].

In addition, current constraints on fixed hardware size and connectivity limit the practical size of problem instances that can be executed on many NPUs and QPUs. There is a significant interest in understanding how larger quantum devices can show speedup and accelerate computing [30, 113, 174], but scaling up these devices to include more qubits or more neurons and denser connections

is a significant device engineering problem [115]. For example, thermal effects present a significant obstacle to scaling quantum devices to large sizes, and recent results posit that this may even inhibit any possible quantum speedup [6]. One workaround is to divide problems that are too large to be directly embedded onto the existing hardware into smaller sub-problems [48], or defining hybrid models. Hybrid classical-quantum solvers have been put forth by D-Wave, in which computationally difficult parts of a computation are off-loaded to a QPU [180]. Hybrid classical-quantum training methods for generative models were derived for the Helmholtz machine of Reference [17] and generally underlie the concept of VQE algorithms [165]. NPUs can operate at room temperature and have been developed to be more scalable, but less emphasis has been placed on how to incorporate them into larger HPC workflows. Currently, they are incorporated into heterogeneous workflows by training ANNs on CPUs and GPUs, prior to deployment on a NPU.

We would like to highlight a specific use case that shows how software has been developed to incorporate DQDs and NPUs into machine learning workflows. The incorporation of DQDs into machine learning faces a significant challenge in the computational cost incurred during gradient evaluation. The PennyLane library developed by Xanadu [18] is developed for machine learning workflows and incorporates auto-differentiation. A computational graph can be constructed that incorporates both classical and quantum nodes. The incorporation of NPUs into machine learning workflows usually requires that an SNN is constructed from a pre-trained ANN. The Whetstone library [234] is targeted toward the integration of SNN into machine learning workflows by both translating an ANN into an SNN and also training the SNN. This library addresses the challenge of how to apply gradient-based training to discrete spike-based computation.

10 GRAPH ALGORITHMS

This review has focused on CSPs and ML as primary targets for acceleration with next-generation processors. These problems have many different facets, which makes them amenable to incorporation into hybrid workflows. However, there are many other problems that can benefit from next-generation processors. The analysis of graphical data is computationally expensive, as most real-world networks can contain millions of vertices and billions of edges. There is a significant need for efficient search and classification algorithms that can be deployed on these higher-dimensional data structures. Many graph algorithms can be implemented using matrix-based computing on GPUs and have been shown to outperform CPUs for certain graph analysis tasks such as triangle counting [262].

One of the compelling graph algorithms for NPUs is shortest-path finding. The shortest-path algorithm as related to spiking neural activity was described in Reference [197], but it has also been recently demonstrated on both real [51] and simulated [228] NPUs. This algorithm takes advantage of the massively parallel nature of NPUs and the natural spike propagation functionality of these systems to perform shortest-path finding. Moreover, these approaches also leverage the on-chip learning capabilites of spiking NPUs in determining the shortest paths.

Another graph analysis problem is community detection, which is both a search and a classification problem. For a graph $\mathcal{G}(V, E)$ the goal of a community detection task is to identify subsets of the vertex set that can be considered "related." This is a problem encountered in many scientific fields where structured, relational data is generated. Many algorithms exist for solving this problem [75, 152, 218], and there is wide use of interacting spin systems in solving community detection problems [21, 110, 206, 207], which have been shown to be resolution-limit free [211].

The synchronicity of spiking neurons with excitatory and inhibitory connections has been exploited to implement label propagation algorithms using leaky integrate and fire neurons or Kuramoto oscillators [54, 202, 203] under global driving and inhibition. A separate spike-based approach to label propagation has been implemented by combing spin-glass dynamics with locally

driven leaky integrate and fire neurons [91, 92], deployed on IBM's TrueNorth [93] and tested on the 128-vertex Girvan-Newman benchmark graph. The algorithm described in Reference [91] relies on a fully connected SNN that could not be directly embedded on the TrueNorth hardware. Instead, the spiking algorithm was embedded onto hardware using a set of sparse sub-trees [93]. The question of how to effectively embed a graph into an SNN to avoid resolution limits remains an open question.

The mathematical construction of graphs and networks make them a natural candidate for spiking neural networks and Ising spin glasses, and many graph algorithms can be constructed as CSPs and deployed on ABDs (see Section 6), though hardware connectivity can limit problem size. These include the traveling salesman problem [175], network flow problems [180], graph coloring problems [48], spanning tree identification [184], and graph isomorphism [268]. In addition, there are a number of studies that show via query complexity proofs that a quantum speedup is possible for several graph problems: triangle finding on sparse graphs [119], triangle finding on dense graphs [32], diameter calculation for network routing [147], and general routing in distributed computing networks [79].

Additional studies have shown how quantum annealers can be used to implement graph partitioning and community detection [189, 253]. In References [236, 253], the D-Wave annealer and a 16-qubit chip from IBM were used to implement community detection and graph partitioning. In Reference [253] the use of multi-node embeddings allowed for the D-Wave annealer to be used for partitioning graphs of up to 1K nodes. For certain graphs the D-Wave out-performed a classical solver by reducing the number of cut-edges between found partitions. In Reference [236], a 16-qubit chip was used in community detection problems on graphs up to 400 vertices by decomposing a larger graph into smaller sub-graphs and using QAOA to iterate over local solutions to find a global solution.

11 CURRENT CHALLENGES AND ADAPTATIONS

There are a number of common challenges to the effective utilization of NPUs and QPUs stemming from the following hardware characteristics: connection sparsity, low-precision computing, and device noise. These three areas affect the hardware devices described above to different degrees, but overall this has led to many different approaches to embedding a problem into a hardware representation to maximize the problem size, the number of gates in a circuit, or network complexity.

For ABDs, DQDs, and custom neuromorphic systems, sparse connectivity at the device level limits the problem size, or the types networks that can be directly embedded onto the hardware. Sparse connectivity can limit the interactions between spins in a quantum annealer, the gates that can be implemented in a digital quantum device, or the communication between neurons in NPUs. For example, IBM's Neurosynaptic System (TrueNorth) [168] and D-Wave's (2X and II) quantum annealers have dense local connectivity and relatively sparse global connectivity. For NISQ devices, noise limits the size of circuits that can be constructed and executed on hardware and has spurred the development of noise mitigation strategies that have shown promising results when applied to shallow circuits [62, 128, 153, 252]. Overall, fixed connectivity and device noise emphasize a need for sparseness in the construction of the model that is deployed on many different types of hardware.

Most physical neuromorphic systems also have low-precision realizations of parameters on neurons and synapses to allow for high efficiency. For example, IBM's TrueNorth has a two-bit weight resolution on its synapses [168]. Intel's Loihi [51] is flexible and supports anywhere from one-bit to nine-bit synaptic weight resolution. These limitations in weight resolution result in adaptations to the training algorithms used to determine synaptic weights in machine learning applications. Rather than using out-of-the-box techniques like back-propagation implementations in TensorFlow, these approaches now have to either be adapted to incorporate restrictions into training

process, or a mapping procedure from high weight resolution networks to lower weight resolution networks must be established [64]. Similarly, the available bits of precision for couplers and biases in the D-Wave annealer affects the complexity of a spin glass that can be deployed on hardware.

A common feature of many near-term and current generation hardware is a strong emphasis on sparse coding or sparse embedding. A common coding feature that has shown promising results on both NPUs and QPUs is sparse coding [214] with locally competitive algorithms [213]. This is a coding technique that is motivated by how the brain encodes information. Sparse coding with local competition has been implemented on Intel's Loihi chip and has been shown to outperform conventional processors for solving LASSO optimization [154]. Sparse coding has also been shown to improve computational time for solving optimization problems on a QPU [182]. The DQDs discussed in this survey are all NISQ devices and the presence of noise and decoherence make the construction of sparse or shallow circuits a necessity.

We close this section with a discussion of a significant bottleneck to the specific application of quantum machine learning on near-term DQDs: gradient-based training. The evaluation of gradients is a significant bottleneck in classical machine learning as well, and a number of quantum algorithms have been developed to reduce the computational cost of evaluating a n-dimensional gradient [81, 124] by utilizing quantum superposition and Quantum Fourier Transforms. However, the circuits that can be evaluated on near-term NISQ hardware are not large enough to implement these algorithms, and the evaluation of an n-dimensional circuit gradient will require $2n$ circuit evaluations. In addition, for circuits trained on near-term NISQ devices, the final state prepared by the circuit is estimated by taking multiple measurements. Shallow circuits can be utilized to evaluate the gradient [153, 223, 224], removing the need for additional qubits, but the reliance on drawing a finite number of samples from the final prepared state introduces the possibility that undersampling states in the final wavefunction can cause training to fail. This is the "barren plateau" problem introduced in Reference [164]. Recent studies have explored how this phenomenon can be avoided by better parameter initialization [85] or by different choice of classical optimizer [16].

12 CONCLUSIONS

We have provided a review of several recent applications of neural networks on next-generation hardware platforms. These examples emphasize that there are both familiar and unfamiliar elements in the design and implementation of applications, such as neural networks or constraint satisfaction problems onto these new systems. A common feature in these designs is the relevance of graphical structures to indicate associations, patterns, and feedback. While the use of these concepts is natural for programming neuromorphic and/or quantum systems, the migration of existing applications is less familiar. However, there are problems that find a natural connection with these data structures, including graph-analytic methods and machine learning tasks. The translation of existing applications to neuromorphic and quantum processing and other future technologies may therefore be expected to proceed in stages with early adoption by select user communities. We expect that success in these areas will set the stage for adoption later by broader user groups. The transition between these stages will depend on the infrastructure that provides support for translating data structure from conventional representations to the structure necessary for next-generation hardware.

There have been growing concerns that emerging computing paradigms, such as neuromorphic computing and quantum computing, will provide such disruptive capability as to undermine existing investments in software infrastructure [37, 44]. However, as the applications presented here show, the disruption lies largely at the interfaces between the specialized methods allocated to these processors and the routines that may make use of them. In this regard, neuromorphic and quantum processing may best befit the accelerator model that has become increasingly popular

with the successful use of GPUs. Designs for these systems emphasize the expectation of extreme heterogeneity in future HPC systems and the need for infrastructure to manage, schedule, and tune these systems. However, we must emphasize that the successful use of neuromorphic and quantum processing to date does not discourage this approach, but rather makes clear that such efforts are worthwhile.

APPENDIX

A HARDWARE CONNECTIVITY OF ABDS AND DQDS

In the main text, we highlight how sparse connectivity can limit the size of models or programs that can be implemented on different next-generation processors. In this Appendix, we include figures of quantum devices that were heavily discussed in the main text: a subgraph of the Chimera layout of the D-Wave 2000Q processor [47] and qubit layouts of two publicly available IBM devices [116]. The Chimera layout (see Figure 1) is characterized by dense local connections but sparse global connections. To increase the connectivity, chains of qubits can be constructed to define a problem

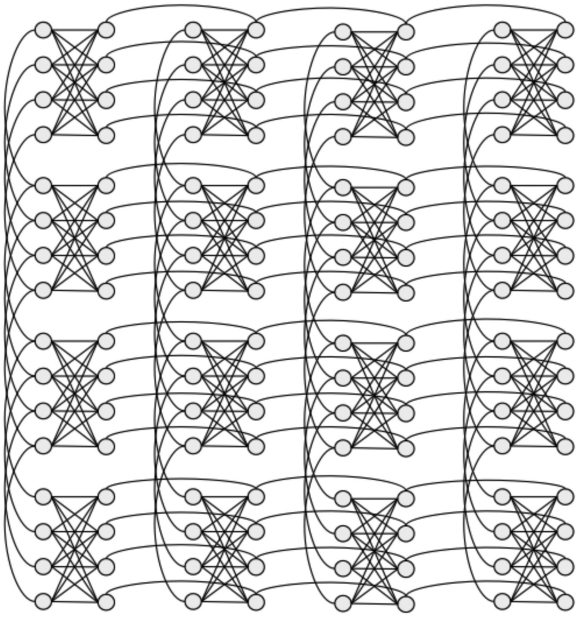

Fig. 1. A subgraph of the Chimera layout. Qubits are arranged in a 4×4 grid of bipartite unit cells with 8 qubits in each unit cell.

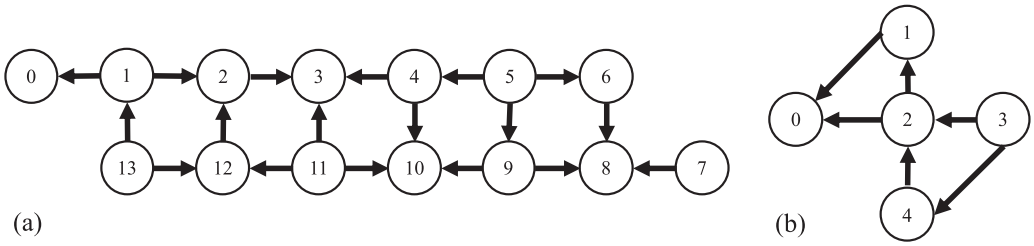

Fig. 2. Layouts of qubits and couplers for two IBM quantum processors. (a) the 14-qubit Melbourne chip has a ladder structure. (b) the 5-qubit Tenerife chip has a bowtie structure.

embedding. The IBM devices (see Figure 2) use superconducting qubits and the orientation of the couplers between adjacent qubits determines which two-qubit gates can be implemented in a circuit.

B NOMENCLATURE

We collect all the acronyms defined in the main text in Table 2 for quick reference.

Table 2. Table of Acronyms

ABD	annealer-based device
ANN	artificial neural network
CNN	convolutional neural network
CPU	central processing unit
CSP	constraint satisfaction problem
DBN	deep belief network
DQD	digital quantum device
GPU	graphical processing unit
HPC	high performance computing
LSTM	long short term memory network
ML	machine learning
MLP	multilevel perceptron
NISQ	noisy intermediate scale quantum (device)
NPU	neural processing unit
OIM	optical Ising machine
QAOA	quantum approximate optimization algorithm
QCL	quantum circuit learning
QML	quantum machine learning
QPU	quantum processing unit
RBM	restricted Boltzmann machine
SAT	constraint satisfiability (computational problem)
SNN	spiking neural network
STDP	spike timing dependent plasticity
SVM	support vector machine
TPU	tensor processing unit

REFERENCES

[1] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. 1985. A learning algorithm for Boltzmann machines. *Cog. Sci.* 9, 1 (1985), 147–169.

[2] Masaharu Adachi and Kazuyuki Aihara. 1997. Associative dynamics in a chaotic neural network. *Neural Netw.* 10, 1 (1997), 83–98.

[3] James B. Aimone, William Severa, and Craig M. Vineyard. 2019. Composing neural algorithms with Fugu. *arXiv preprint arXiv:1905.12130* (2019).

[4] Tameem Albash and Daniel A. Lidar. 2018. Adiabatic quantum computation. *Rev. Mod. Phys.* 90 (Jan. 2018), 015002. Issue 1. DOI: https://doi.org/10.1103/RevModPhys.90.015002

[5] Tameem Albash and Daniel A. Lidar. 2018. Demonstration of a scaling advantage for a quantum annealer over simulated annealing. *Phys. Rev. X* 8 (July 2018), 031016. Issue 3. DOI: https://doi.org/10.1103/PhysRevX.8.031016

[6] Tameem Albash, Victor Martin-Mayor, and Itay Hen. 2017. Temperature scaling law for quantum annealing optimizers. *Phys. Rev. Lett.* 119, 11 (2017), 110502.

[7] Behrooz Kamary Aliabadi, Claude Berrou, Vincent Gripon, and Xiaoran Jiang. 2014. Storing sparse messages in networks of neural cliques. *IEEE Transactions on Neural Networks and Learning Systems* 25, 5 (2014), 980–989.

[8] Shun-Ichi Amari. 1989. Characteristics of sparsely encoded associative memory. *Neural Networks* 2, 6 (1989), 451–457.

[9] Daniel J. Amit, Hanoch Gutfreund, and Haim Sompolinsky. 1985. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Phys. Rev. Lett.* 55, 14 (1985), 1530.

[10] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. 2016. Deep speech 2 : End-to-end speech recognition in English and Mandarin. In *Proceedings of the 33rd International Conference on Machine Learning*), Maria Florina Balcan and Kilian Q. Weinberger (Eds.), Vol. 48. PMLR, New York, New York, 173–182. Retrieved from http://proceedings.mlr.press/v48/amodei16.html.

[11] Maliheh Aramon, Gili Rosenberg, Elisabetta Valiante, Toshiyuki Miyazawa, Hirotaka Tamura, and Helmut Katzgrabeer. 2019. Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. *Front. Phys.* 7 (2019), 48.

[12] John V. Arthur, Paul A. Merolla, Filipp Akopyan, Rodrigo Alvarez, Andrew Cassidy, Shyamal Chandra, Steven K. Esser, Nabil Imam, William Risk, Daniel B. D. Rubin, et al. 2012. Building block of a programmable neuromorphic substrate: A digital neurosynaptic core. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'12)*. IEEE, 1–8.

[13] Demian A. Battaglia, Giuseppe E. Santoro, and Erio Tosatti. 2005. Optimization by quantum annealing: Lessons from hard satisfiability problems. *Phys. Rev. E* 71 (June 2005), 066707. Issue 6. DOI: https://doi.org/10.1103/PhysRevE.71.066707

[14] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940* (2016).

[15] Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, and Alejandro Perdomo-Ortiz. 2019. A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quant. Inf.* 5, 1 (2019), 45.

[16] Marcello Benedetti, Edward Grant, Leonard Wossnig, and Simone Severini. 2019. Adversarial quantum circuit learning for pure state approximation. *New J. Phys.* 21, 4 (2019), 043023.

[17] Marcello Benedetti, John Realpe-Gómez, and Alejandro Perdomo-Ortiz. 2018. Quantum-assisted Helmholtz machines: A quantum–classical deep learning framework for industrial datasets in near-term devices. *Quant. Sci. Technol.* 3, 3 (2018), 034007.

[18] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, and N. Killoran. 2018. PennyLane: Automatic differentiation of hybrid quantum-classical computations. *ArXiv e-prints* (Nov. 2018). arxiv:quant-ph/1811.04968

[19] Hannes Bernien, Sylvain Schwartz, Alexander Keesling, Harry Levine, Ahmed Omran, Hannes Pichler, Soonwon Choi, Alexander S. Zibrov, Manuel Endres, Markus Greiner, et al. 2017. Probing many-body dynamics on a 51-atom quantum simulator. *Nature* 551, 7682 (2017), 579.

[20] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2017. Quantum machine learning. *Nature* 549, 7671 (2017), 195.

[21] Marcelo Blatt, Shai Wiseman, and Eytan Domany. 1996. Superparamagnetic clustering of data. *Phys. Rev. Lett.* 76 (Apr. 1996), 3251–3254. Issue 18. DOI: https://doi.org/10.1103/PhysRevLett.76.3251

[22] Jason W. Bohland and Ali A. Minai. 2001. Efficient associative memory using small-world architecture. *Neurocomputing* 38 (2001), 489–496.

[23] Sander M. Bohte, Joost N. Kok, and Han La Poutre. 2002. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* 48, 1 (2002), 17–37.

[24] Tomas Boothby, Andrew D. King, and Aidan Roy. 2016. Fast clique minor generation in Chimera qubit connectivity graphs. *Quant. Inf. Proc.* 15, 1 (2016), 495–508.

[25] J. Boyd. 2018. Silicon chip delivers quantum speeds. *IEEE Spectrum* 55, 7 (July 2018), 10–11. DOI: https://doi.org/10.1109/MSPEC.2018.8389173

[26] Sergey Bravyi, David Gosset, and Robert Koenig. 2018. Quantum advantage with shallow circuits. *Science* 362, 6412 (2018), 308–311.

[27] Keith A. Britt and Travis S. Humble. 2017. High-performance computing with quantum processing units. *ACM J. Emerg. Technol. Comput. Syst.* 13, 3 (2017), 39.

[28] Keith A Britt and Travis S Humble. 2017. High-performance computing with quantum processing units. *ACM J. Emerg. Technol. Comput. Syst.* 13, 3 (2017), 39.

[29] K. A. Britt, F. A. Mohiyaddin, and T. S. Humble. 2017. Quantum accelerators for high-performance computing systems. In *Proceedings of the IEEE International Conference on Rebooting Computing (ICRC'17)*. 1–7. DOI: https://doi.org/10.1109/ICRC.2017.8123664

[30] K. A. Britt, F. A. Mohiyaddin, and T. S. Humble. 2017. Quantum accelerators for high-performance computing systems. In *Proceedings of the IEEE International Conference on Rebooting Computing (ICRC'17)*. 1–7. DOI: https://doi.org/10.1109/ICRC.2017.8123664

[31] Colin D. Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M. Sage. 2019. Trapped-ion quantum computing: Progress and challenges. *Appl. Phys. Rev.* 6, 2 (2019), 021314.

[32] Harry Buhrman, Christoph Durr, Mark Heiligman, Peter Hoyer, Frédéric Magniez, Miklos Santha, and Ronald De Wolf. 2001. Quantum algorithms for element distinctness. In *Proceedings of the 16th IEEE Conference on Computational Complexity*. IEEE, 131–137.

[33] Harald Burgsteiner, Mark Kröll, Alexander Leopold, and Gerald Steinbauer. 2007. Movement prediction from real-world images using a liquid state machine. *Appl. Intell.* 26, 2 (2007), 99–109.

[34] Jeremie Cabessa and Hava T. Siegelmann. 2014. The super-Turing computational power of plastic recurrent neural networks. *Int. J. Neural Syst.* 24, 8 (2014), 1450029.

[35] Jun Cai, William G. Macready, and Aidan Roy. 2014. A practical heuristic for finding graph minors. *arXiv preprint arXiv:1406.2741* (2014).

[36] Andrea Calimera, Enrico Macii, and Massimo Poncino. 2013. The human brain project and neuromorphic computing. *Funct. Neurol.* 28, 3 (2013), 191.

[37] J. Carballo, W. J. Chan, P. A. Gargini, A. B. Kahng, and S. Nath. 2014. ITRS 2.0: Toward a re-framing of the semiconductor technology roadmap. In *Proceedings of the IEEE 32nd International Conference on Computer Design (ICCD'14)*. 139–146. DOI: https://doi.org/10.1109/ICCD.2014.6974673

[38] Andrew S. Cassidy, Paul Merolla, John V. Arthur, Steve K. Esser, Bryan Jackson, Rodrigo Alvarez-Icaza, Pallab Datta, Jun Sawada, Theodore M. Wong, Vitaly Feldman, et al. 2013. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'13)*. IEEE, 1–10.

[39] Gangotree Chakma, Md Musabbir Adnan, Austin R. Wyer, Ryan Weiss, Catherine D. Schuman, and Garrett S. Rose. 2018. Memristive mixed-signal neuromorphic systems: Energy-efficient learning at the circuit-level. *IEEE J. Emerg. Select. Topics Circ. Syst.* 8, 1 (2018), 125–136.

[40] Chia Cheng Chang, Arjun Gambhir, Travis S. Humble, and Shigetoshi Sota. 2019. Quantum annealing for systems of polynomial equations. *Sci. Rep.* 9, 1 (2019), 10258.

[41] Vicky Choi. 2008. Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. *Quant. Inf. Proc.* 7, 5 (2008), 193–209.

[42] Vicky Choi. 2011. Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design. *Quant. Inf. Proc.* 10, 3 (2011), 343–353.

[43] William R. Clements, Jelmer J. Renema, Y. Henry Wen, Helen M. Chrzanowski, W. Steven Kolthammer, and Ian A. Walmsley. 2017. Gaussian optical Ising machines. *Phys. Rev. A* 96, 4 (Oct. 2017). DOI: https://doi.org/10.1103/physreva.96.043850

[44] Thomas M. Conte, Erik P. DeBenedictis, Paolo A. Gargini, and Elie Track. 2017. Rebooting computing: The road ahead. *Computer* 50, 1 (2017), 20–29. DOI: https://doi.org/doi.ieeecomputersociety.org/10.1109/MC.2017.8

[45] Nigel Crook, Wee Jin Goh, and Mohammad Hawarat. 2007. Pattern recall in networks of chaotic neurons. *BioSystems* 87, 2–3 (2007), 267–274.

[46] George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Math. Contr. Signals Syst.* 2, 4 (1989), 303–314.

[47] D-WAVE. 2018. D-WAVE Publications. Retrieved from http://www.dwavesys.com/resources/publications.

[48] E. D. Dahl. 2013. *Programming with D-Wave: Map Coloring Problem*. Technical Report. D-Wave Systems. Retrieved from https://www.dwavesys.com/resources/publications.

[49] Pierre-Luc Dallaire-Demers and Nathan Killoran. 2018. Quantum generative adversarial networks. *Phys. Rev. A* 98 (July 2018), 012324. Issue 1. DOI: https://doi.org/10.1103/PhysRevA.98.012324

[50] Srinjoy Das, Bruno Umbria Pedroni, Paul Merolla, John Arthur, Andrew S. Cassidy, Bryan L. Jackson, Dharmendra Modha, Gert Cauwenberghs, and Ken Kreutz-Delgado. 2015. Gibbs sampling with low-power spiking digital neurons. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'15)*. IEEE, 2704–2707.

[51] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 1 (2018), 82–99.

[52] Andrew P. Davison, Daniel Brüderle, Jochen M. Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. 2009. PyNN: A common interface for neuronal network simulators. *Front. Neuroinf.* 2 (2009), 11.

[53] Ricardo de Azambuja, Frederico B. Klein, Martin F. Stoelen, Samantha V. Adams, and Angelo Cangelosi. 2016. Graceful degradation under noise on brain inspired robot controllers. In *Proceedings of the International Conference on Neural Information Processing*. Springer, 195–204.

[54] João Eliakin Mota de Oliveira and Marcos G. Quiles. 2014. Community detection in complex networks using coupled Kuramoto oscillators. In *Proceedings of the 14th International Conference on Computational Science and Its Applications (ICCSA'14)*. IEEE, 85–90.

[55] Shantanu Debnath, Norbert M. Linke, Caroline Figgatt, Kevin A. Landsman, Kevin Wright, and Christopher Monroe. 2016. Demonstration of a small programmable quantum computer with atomic qubits. *Nature* 536, 7614 (2016), 63.

[56] C. L. Degen, F. Reinhard, and P. Cappellaro. 2017. Quantum sensing. *Rev. Mod. Phys.* 89, 3 (2017), 035002.

[57] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. 2016. What is the computational value of finite-range tunneling? *Phys. Rev. X* 6, 3 (2016), 031015.

[58] N. G. Dickson, M. W. Johnson, M. H. Amin, R. Harris, F. Altomare, A. J. Berkley, P. Bunyk, J. Cai, E. M. Chapple, P. Chavez, F. Cioata, T. Cirip, P. deBuen, M. Drew-Brook, C. Enderud, S. Gildert, F. Hamze, J. P. Hilton, E. Hoskinson, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Lanting, T. Mahon, R. Neufeld, T. Oh, I. Perminov, C. Petroff, A. Przybysz, C. Rich, P. Spear, A. Tcaciuc, M. C. Thom, E. Tolkacheva, S. Uchaikin, J. Wang, A. B. Wilson, Z. Merali, and G. Rose. 2013. Thermally assisted quantum annealing of a 16-qubit problem. *Nat. Commun.* 4 (21 05 2013), 1903. Retrieved from http://dx.doi.org/10.1038/ncomms2920.

[59] Peter U. Diehl, Guido Zarrella, Andrew Cassidy, Bruno U. Pedroni, and Emre Neftci. 2016. Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware. In *Proceedings of the IEEE International Conference on Rebooting Computing (ICRC'16)*. IEEE, 1–8.

[60] Adam Douglass, Andrew D. King, and Jack Raymond. 2015. Constructing SAT filters with a quantum annealer. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing*. Springer, 104–120.

[61] Vedran Dunjko and Hans J. Briegel. 2018. Machine learning & artificial intelligence in the quantum domain: A review of recent progress. *Rep. Prog. Phys.* 81, 7 (2018), 074001.

[62] Suguru Endo, Simon C. Benjamin, and Ying Li. 2018. Practical quantum error mitigation for near-future applications. *Phys. Rev. X* 8, 3 (2018), 031027.

[63] Steve K. Esser, Alexander Andreopoulos, Rathinakumar Appuswamy, Pallab Datta, Davis Barch, Arnon Amir, John Arthur, Andrew Cassidy, Myron Flickner, Paul Merolla, et al. 2013. Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'13)*. IEEE, 1–10.

[64] Steve K. Esser, Rathinakumar Appuswamy, Paul Merolla, John V. Arthur, and Dharmendra S. Modha. 2015. Backpropagation for energy-efficient neuromorphic computing. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 1117–1125.

[65] Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, et al. 2016. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proc. Nat. Acad. Sci.* (2016), 201604850.

[66] Zhe Fan, Feng Qiu, Arie Kaufman, and Suzanne Yoakum-Stover. 2004. GPU cluster for high performance computing. In *Proceedings of the ACM/IEEE Supercomputing Conference*. IEEE, 47–47.

[67] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028* (2014).

[68] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. 2001. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science* 292, 5516 (2001), 472–475.

[69] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Hartmut Neven. 2017. Quantum algorithms for fixed qubit architectures. *arXiv preprint arXiv:1703.06199* (2017).

[70] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. 2000. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106* (2000).

[71] Edward Farhi and Aram W. Harrow. 2016. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv preprint arXiv:1602.07674* (2016).

[72] Edward Farhi and Hartmut Neven. 2018. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002* (2018).

[73] Ronald A. Fisher. 1936. The use of multiple measurements in taxonomic problems. *Ann. Eug.* 7, 2 (1936), 179–188.

[74] Gabriel Andrés Fonseca Guerra and Steve B. Furber. 2017. Using stochastic spiking neural networks on SpiNNaker to solve constraint satisfaction problems. *Front. Neurosci.* 11 (2017), 714.

Q4

[75] Santo Fortunato. 2010. Community detection in graphs. *Phys. Rep.* 486, 3 (2010), 75–174.
[76] Keisuke Fujii and Kohei Nakajima. 2017. Harnessing disordered-ensemble quantum dynamics for machine learning. *Phys. Rev. Appl.* 8, 2 (2017), 024030.
[77] Fujitsu. 2018. Fujitsu Quantum-inspired Computing Digital Annealer. *Retrieved from http://www.fujitsu.com/global/digitalannealer/*.
[78] Steve B. Furber, Francesco Galluppi, Steve Temple, and Luis A. Plana. 2014. The SpiNNaker project. *Proc. IEEE* 102, 5 (2014), 652–665.
[79] François Le Gall, Harumichi Nishimura, and Ansis Rosmanis. 2018. Quantum advantage for the LOCAL model in distributed computing. *arXiv preprint arXiv:1810.10838* (2018).
[80] Sanjib Ghosh, Andrzej Opala, Michał Matuszewski, Tomasz Paterek, and Timothy C. H. Liew. 2019. Quantum reservoir processing. *npj Quant. Inf.* 5, 1 (2019), 35.
[81] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. 2019. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 1425–1444.
[82] D. Golomb, N. Rubin, and Haim Sompolinsky. 1990. Willshaw model: Associative memory with sparse coding and low firing rates. *Phys. Rev. A* 41, 4 (1990), 1843.
[83] Timothy D. Goodrich, Blair D. Sullivan, and Travis S. Humble. 2018. Optimizing adiabatic quantum program compilation using a graph-theoretic framework. *Quant. Inf. Proc.* 17, 5 (2018), 118.
[84] Edward Grant, Marcello Benedetti, Shuxiang Cao, Andrew Hallam, Joshua Lockhart, Vid Stojevic, Andrew G. Green, and Simone Severini. 2018. Hierarchical quantum classifiers. *npj Quant. Inf.* 4, 1 (2018), 65.
[85] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. 2019. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *arXiv preprint arXiv:1903.05076* (2019).
[86] Vincent Gripon and Claude Berrou. 2011. Sparse neural networks with large learning diversity. *IEEE Trans. Neural Netw.* 22, 7 (2011), 1087–1096.
[87] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th ACM Symposium on Theory of Computing*. ACM, 212–219.
[88] Ryan Hamerly, Takahiro Inagaki, Peter L. McMahon, Davide Venturelli, Alireza Marandi, Tatsuhiro Onodera, Edwin Ng, Carsten Langrock, Kensuke Inaba, Toshimori Honjo, Koji Enbutsu, Takeshi Umeki, Ryoichi Kasahara, Shoko Utsunomiya, Satoshi Kako, Ken ichi Kawarabayashi, Robert L. Byer, Martin M. Fejer, Hideo Mabuchi, Dirk Englund, Eleanor Rieffel, Hiroki Takesue, and Yoshihisa Yamamoto. [n.d.]. Experimental investigation of performance differences between Coherent Ising Machines and a quantum annealer. ([n. d.]). arXiv:quant-ph/1805.05217v2.
[89] Kathleen E. Hamilton, Eugene F. Dumitrescu, and Raphael C. Pooser. 2019. Generative model benchmarks for superconducting qubits. *Phys. Rev. A* 99, 6 (2019), 062323.
[90] Kathleen E. Hamilton and Travis S. Humble. 2017. Identifying the minor set cover of dense connected bipartite graphs via random matching edge sets. *Quant. Inf. Proc.* 16, 4 (2017), 94.
[91] Kathleen E. Hamilton and Travis S. Humble. 2018. Spiking spin-glass models for label propagation and community detection. *arXiv preprint arXiv:1801.03571* (2018).
[92] Kathleen E. Hamilton, Neena Imam, and Travis S. Humble. 2017. Community detection with spiking neural networks for neuromorphic hardware. In *Proceedings of the Neuromorphic Computing Symposium*. ACM, 9.
[93] Kathleen E. Hamilton, Neena Imam, and Travis S. Humble. 2018. Sparse hardware embedding of spiking neuron systems for community detection. *J. Emerg. Technol. Comput. Syst.* 14, 4, Article 40 (Nov. 2018), 13 pages. DOI: https://doi.org/10.1145/3223048
[94] Kathleen E. Hamilton, Catherine D. Schuman, Steven R. Young, Neena Imam, and Travis S. Humble. 2018. Neural networks and graph algorithms with next-generation processors. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW'18)*. IEEE, 1194–1203.
[95] Yoshitaka Haribara, Shoko Utsunomiya, and Yoshihisa Yamamoto. 2016. Computational principle and performance evaluation of coherent Ising machine based on degenerate optical parametric oscillator network. *Entropy* 18, 4 (Apr. 2016), 151. DOI: https://doi.org/10.3390/e18040151
[96] R. Harris, Y. Sato, A. J. Berkley, M. Reis, F. Altomare, M. H. Amin, K. Boothby, P. Bunyk, C. Deng, C. Enderud, et al. 2018. Phase transitions in a programmable quantum spin glass simulator. *Science* 361, 6398 (2018), 162–165.
[97] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. 2019. Supervised learning with quantum-enhanced feature spaces. *Nature* 567, 7747 (2019), 209.
[98] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*. 1026–1034.
[99] Itay Hen and Federico M. Spedalieri. 2016. Quantum annealing for constrained optimization. *Phys. Rev. Appl.* 5, 3 (2016), 034007.

[100] Greg Henry, Ping Tak Peter Tang, and Alexander Heinecke. 2019. Leveraging the bfloat16 artificial intelligence datatype for higher-precision computations. *CoRR* abs/1904.06376 (2019).
[101] John Hertz, Anders Krogh, and Richard G. Palmer. 1991. *Introduction to the Theory of Neural Computation*. Santa Fe Institute Studies in the Sciences of Complexity, Vol. 1. Addison-Wesley.
[102] Christopher J. Hillar and Lek-Heng Lim. 2013. Most tensor problems are NP-hard. *J. ACM* 60, 6, Article 45 (Nov. 2013), 39 pages. DOI: https://doi.org/10.1145/2512329
[103] Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computat.* 14, 8 (2002), 1771–1800.
[104] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computat.* 18, 7 (2006), 1527–1554.
[105] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computat.* 9, 8 (1997), 1735–1780. DOI: https://doi.org/10.1162/neco.1997.9.8.1735
[106] John J. Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci.* 79, 8 (1982), 2554–2558.
[107] John J. Hopfield. 1984. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Nat. Acad. Sci.* 81, 10 (1984), 3088–3092.
[108] John J. Hopfield and David W. Tank. 1985. Neural computation of decisions in optimization problems. *Biol. Cyber.* 52, 3 (1985), 141–152.
[109] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural Netw.* 2, 5 (1989), 359–366.
[110] Dandan Hu, Peter Ronhovde, and Zohar Nussinov. 2012. Phase transitions in random Potts systems and the community detection problem: Spin-glass type and dynamic perspectives. *Philos. Mag.* 92, 4 (2012), 406–445.
[111] Gary B. Huang, Honglak Lee, and Erik Learned-Miller. 2012. Learning hierarchical representations for face verification with convolutional deep belief networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12)*. IEEE, 2518–2525.
[112] Travis S. Humble, Alex J. McCaskey, Ryan S. Bennink, Jay Jay Billings, E. F. D'Azevedo, Blair D. Sullivan, Christine F. Klymko, and Hadayat Seddiqi. 2014. An integrated programming and development environment for adiabatic quantum optimization. *Computat. Sci. Disc.* 7, 1 (2014), 015006.
[113] Travis S. Humble, Alexander J. McCaskey, Jonathan Schrock, Hadayat Seddiqi, Keith A. Britt, and Neena Imam. 2016. Performance models for split-execution computing systems. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops*. IEEE, 545–554.
[114] Travis S. Humble, Ronald J. Sadlier, and Keith A. Britt. 2018. Simulated execution of hybrid quantum computing systems. In *Proceedings of the Quantum Information Science, Sensing, and Computation X*, Vol. 10660. International Society for Optics and Photonics, 1066002.
[115] Travis S. Humble, Himanshu Thapliyal, Edgard Munoz-Coreas, Fahd A. Mohiyaddin, and Ryan S. Bennink. 2019. Quantum computing circuits and devices. *IEEE Des. Test* 36, 3 (2019), 69–94.
[116] IBM. 2018. IBM Q-Experience. Retrieved from http://www.research.ibm.com/ibm-q/.
[117] IBM. 2018. Qiskit Aqua. Retrieved from https://qiskit.org/aqua.
[118] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu, O. Tadanaga, H. Takenouchi, K. Aihara, K. I. Kawarabayashi, K. Inoue, S. Utsunomiya, and H. Takesue. 2016. A coherent Ising machine for 2000-node optimization problems. *Science* 354, 6312 (Oct. 2016), 603–606. DOI: https://doi.org/10.1126/science.aah4243
[119] Taisuke Izumi and François Le Gall. 2017. Triangle finding and listing in CONGEST networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*. ACM, 381–389.
[120] Shuxian Jiang, Keith A. Britt, Alexander J. McCaskey, Travis S. Humble, and Sabre Kais. 2018. Quantum annealing for prime factorization. *Sci. Rep.* 8, 1 (2018), 17667.
[121] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose. 2011. Quantum annealing with manufactured spins. *Nature* 473, 7346 (12 05 2011), 194–198. Retrieved from http://dx.doi.org/10.1038/nature10012.
[122] Zeno Jonke, Stefan Habenschuss, and Wolfgang Maass. 2016. Solving constraint satisfaction problems with networks of spiking neurons. *Front. Neurosci.* 10 (2016), 118. DOI: https://doi.org/10.3389/fnins.2016.00118
[123] Steven Jordan. 2018. Quantum Algorithm Zoo. Retrieved from https://math.nist.gov/quantum/zoo/.
[124] Stephen P. Jordan. 2005. Fast quantum algorithm for numerical gradient estimation. *Phys. Rev. Lett.* 95, 5 (2005), 050501.
[125] N. Jouppi, C. Young, N. Patil, and D. Patterson. 2018. Motivation for and evaluation of the first tensor processing unit. *IEEE Micro* 38, 3 (May 2018), 10–19. DOI: https://doi.org/10.1109/MM.2018.032271057

[126] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th International Symposium on Computer Architecture (ISCA'17)*. ACM, New York, NY, 1–12. DOI: https://doi.org/10.1145/3079856.3080246

[127] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. 2017. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* 549, 7671 (2017), 242.

[128] Abhinav Kandala, Kristan Temme, Antonio D. Córcoles, Antonio Mezzacapo, Jerry M. Chow, and Jay M. Gambetta. 2019. Error mitigation extends the computational reach of a noisy quantum processor. *Nature* 567, 7749 (2019), 491.

[129] Pentti Kanerva. 1988. *Sparse Distributed Memory*. The MIT Press.

[130] Nikola Kasabov, Kshitij Dhoble, Nuttapod Nuntalid, and Giacomo Indiveri. 2013. Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition. *Neural Netw.* 41 (2013), 188–201.

[131] Helmut G. Katzgraber, Firas Hamze, and Ruben S. Andrist. 2014. Glassy chimeras could be blind to quantum speedup: Designing better benchmarks for quantum annealing machines. *Phys. Rev. X* 4 (Apr. 2014), 021008. Issue 2. DOI: https://doi.org/10.1103/PhysRevX.4.021008

[132] J. Kelly, Z. Chen, B. Chiaro, B. Foxen, J. Martinis, and Google Quantum Hardware Team. 2019. Operating and characterizing of a 72 superconducting qubit processor "Bristlecone": Part 1. In *APS Meet. Abstr.* Article A42.002, A42.002 pages.

[133] Volodymyr V. Kindratenko, Jeremy J. Enos, Guochun Shi, Michael T. Showerman, Galen W. Arnold, John E. Stone, James C. Phillips, and Wen-mei Hwu. 2009. GPU clusters for high-performance computing. In *Proceedings of the IEEE International Conference on Cluster Computing and Workshops (CLUSTER'09)*. IEEE, 1–8.

[134] Andrew D. King, William Bernoudy, James King, Andrew J. Berkley, and Trevor Lanting. [n.d.]. Emulating the coherent Ising machine with a mean-field algorithm. arXiv:quant-ph/1806.08422v1.

[135] Andrew D. King, Juan Carrasquilla, Jack Raymond, Isil Ozfidan, Evgeny Andriyash, Andrew Berkley, Mauricio Reis, Trevor Lanting, Richard Harris, Fabio Altomare, et al. 2018. Observation of topological phenomena in a programmable lattice of 1,800 qubits. *Nature* 560, 7719 (2018), 456.

[136] Andrew D. King, Trevor Lanting, and Richard Harris. 2015. Performance of a quantum annealer on range-limited constraint satisfaction problems. *arXiv preprint arXiv:1502.02098* (2015).

[137] James King, Sheir Yarkoni, Jack Raymond, Isil Ozfidan, Andrew D. King, Mayssam Mohammadi Nevisi, Jeremy P. Hilton, and Catherine C. McGeoch. 2017. Quantum annealing amid local ruggedness and global frustration. *arXiv preprint arXiv:1701.04579* (2017).

[138] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220, 4598 (1983), 671–680.

[139] Christine Klymko, Blair D. Sullivan, and Travis S. Humble. 2014. Adiabatic quantum programming: Minor embedding with hard faults. *Quant. Inf. Proc.* 13, 3 (2014), 709–729.

[140] Bart Kosko. 1988. Bidirectional associative memories. *IEEE Trans. Syst. Man Cyber.* 18, 1 (1988), 49–60.

[141] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P. Orlando, Simon Gustavsson, and William D. Oliver. 2019. A quantum engineer's guide to superconducting qubits. *Appl. Phys. Rev.* 6, 2 (2019), 021318.

[142] Mario Krenn, Mehul Malik, Thomas Scheidl, Rupert Ursin, and Anton Zeilinger. 2016. Quantum communication with photons. In *Optics in Our Time*. Springer, 455–482.

[143] Dhireesha Kudithipudi, Qutaiba Saleh, Cory Merkel, James Thesing, and Bryant Wysocki. 2015. Design and analysis of a neuromemristive reservoir computing architecture for biosignal processing. *Front. Neurosci.* 9 (2015).

[144] Manjari S. Kulkarni and Christof Teuscher. 2012. Memristor-based reservoir computing. In *Proceedings of the IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH'12)*. IEEE, 226–232.

[145] Vipin Kumar. 1992. Algorithms for constraint-satisfaction problems: A survey. *AI Mag.* 13, 1 (1992), 32.

[146] Christopher R. Laumann, Roderich Moessner, Antonello Scardicchio, and S. L. Sondhi. 2015. Quantum annealing: The fastest route to quantum computation? *Europ. Phys. J. Spec. Top.* 224, 1 (2015), 75–88.

[147] François Le Gall and Frédéric Magniez. 2018. Sublinear-time quantum computation of the diameter in CONGEST networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*. ACM, 337–346.
[148] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.
[149] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[150] D.-L. Lee. 2006. Improvements of complex-valued Hopfield associative memory by using generalized projection rules. *IEEE Trans. Neural Netw.* 17, 5 (2006), 1341–1347.
[151] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th International Conference on Machine Learning*. ACM, 609–616.
[152] Jure Leskovec, Kevin J. Lang, and Michael Mahoney. 2010. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th International Conference on World Wide Web*. ACM, 631–640.
[153] Ying Li and Simon C. Benjamin. 2017. Efficient variational quantum simulator incorporating active error minimization. *Phys. Rev. X* 7, 2 (2017), 021050.
[154] Chit-Kwan Lin, Andreas Wild, Gautham Chinya, Mike Davies, Narayan Srinivasa, Dan Lavery, and Hong Wang. 2018. Programming spiking neural networks on Intel Loihi. *Computer* (2018).
[155] Norbert M. Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A. Landsman, Kenneth Wright, and Christopher Monroe. 2017. Experimental comparison of two quantum computing architectures. *Proc. Nat. Acad. Sci.* (2017), 201618020.
[156] Jin-Guo Liu and Lei Wang. 2018. Differentiable learning of quantum circuit born machines. *Phys. Rev. A* 98, 6 (2018), 062324.
[157] Seth Lloyd and Christian Weedbrook. 2018. Quantum generative adversarial learning. *Phys. Rev. Lett.* 121 (July 2018), 040502. Issue 4. DOI: https://doi.org/10.1103/PhysRevLett.121.040502
[158] Andrew Lucas. 2014. Ising formulations of many NP problems. *Front. Phys.* 2 (2014), 5.
[159] Wolfgang Maass. 1997. Networks of spiking neurons: The third generation of neural network models. *Neural Netw.* 10, 9 (1997), 1659–1671.
[160] Alan K. Mackworth and Eugene C. Freuder. 1985. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artif. Intelligence* 25, 1 (1985), 65–74.
[161] Alireza Marandi, Zhe Wang, Kenta Takata, Robert L. Byer, and Yoshihisa Yamamoto. 2014. Network of time-multiplexed optical parametric oscillators as a coherent Ising machine. *Nat. Photon.* 8, 12 (Oct. 2014), 937–942. DOI: https://doi.org/10.1038/nphoton.2014.249
[162] Margaret Martonosi and Martin Roetteler. 2019. Next steps in quantum computing: Computer science's role. *arXiv preprint arXiv:1903.10541* (2019).
[163] A. J. McCaskey, E. F. Dumitrescu, D. Liakh, M. Chen, W. Feng, and T. S. Humble. 2018. A language and hardware independent approach to quantum–classical computing. *SoftwareX* 7 (2018), 245–254.
[164] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. 2018. Barren plateaus in quantum neural network training landscapes. *Nat. Commun.* 9, 1 (2018), 4812.
[165] Jarrod R. McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. 2016. The theory of variational hybrid quantum-classical algorithms. *New J. Phys.* 18, 2 (2016), 023023.
[166] Robert J. McEliece, Edward C. Posner, Eugene R. Rodemich, and Santosh S. Venkatesh. 1987. The capacity of the Hopfield associative memory. *IEEE Trans. Inf. Theor.* 33, 4 (1987), 461–482.
[167] Peter L. McMahon, Alireza Marandi, Yoshitaka Haribara, Ryan Hamerly, Carsten Langrock, Shuhei Tamate, Takahiro Inagaki, Hiroki Takesue, Shoko Utsunomiya, Kazuyuki Aihara, Robert L. Byer, M. M. Fejer, Hideo Mabuchi, and Yoshihisa Yamamoto. 2016. A fully programmable 100-spin coherent Ising machine with all-to-all connections. *Science* 354, 6312 (Oct. 2016), 614–617. DOI: https://doi.org/10.1126/science.aah5178
[168] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 6197 (2014), 668–673.
[169] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. 2018. Quantum circuit learning. *Phys. Rev. A* 98 (Sept. 2018), 032309. Issue 3. DOI: https://doi.org/10.1103/PhysRevA.98.032309
[170] Abdel-rahman Mohamed, Tara N. Sainath, George Dahl, Bhuvana Ramabhadran, Geoffrey E. Hinton, and Michael A. Picheny. 2011. Deep belief networks using discriminative features for phone recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'11)*. IEEE, 5060–5063.
[171] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme. 2018. Quantum optimization using variational algorithms on near-term quantum devices. *Quant. Sci. Technol.* 3, 3 (July 2018), 030503. DOI: https://doi.org/10.1088/2058-9565/aab822 arxiv:quant-ph/1710.01022

Q7

[172] Ashley Montanaro. 2016. Quantum algorithms: An overview. *npj Quant. Inf.* 2 (2016), 15023.
[173] C. Moore and S. Mertens. 2011. *The Nature of Computation*. OUP Oxford. Retrieved from https://books.google.com/books?id=5M8le2uAIC8C.
[174] Andrea Morello. 2018. What would you do with 1000 qubits? *Quant. Sci. Technol.* 3, 3 (2018), 030201. Retrieved from http://stacks.iop.org/2058-9565/3/i=3/a=030201.
[175] Dominic J. Moylett, Noah Linden, and Ashley Montanaro. 2017. Quantum speedup of the traveling-salesman problem for bounded-degree graphs. *Phys. Rev. A* 95 (Mar. 2017), 032323. Issue 3. DOI: https://doi.org/10.1103/PhysRevA.95.032323
[176] Joseph S. Najem, Graham J. Taylor, Ryan J. Weiss, Md Sakib Hasan, Garrett Rose, Catherine D. Schuman, Alex Belianinov, C. Patrick Collier, and Stephen A. Sarles. 2018. Memristive ion channel-doped biomembranes as synaptic mimics. *ACS Nano* (2018).
[177] Thomas Natschläger, Wolfgang Maass, and Henry Markram. 2002. The "liquid computer": A novel strategy for real-time computing on time series. *Spec. Issue Found. Inf. Proc. of TELEMATIK* 8, LNMC-ARTICLE-2002-005 (2002), 39–43.
[178] Emre O. Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. 2017. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Front. Neurosci.* 11 (2017), 324.
[179] Nengo. 2018. The Nengo Neural Simulator. Retrieved from https://www.nengo.ai/.
[180] Florian Neukart, Gabriele Compostella, Christian Seidel, David Von Dollen, Sheir Yarkoni, and Bob Parney. 2017. Optimizing traffic flow using quantum annealing and classical machine learning. *arXiv preprint arXiv:1708.01625* (2017).
[181] Hartmut Neven, Geordie Rose, and William G. Macready. 2008. Image recognition with an adiabatic quantum computer I. Mapping to quadratic unconstrained binary optimization. *arXiv preprint arXiv:0804.4457* (2008).
[182] N. T. T. Nguyen, A. E. Larson, and G. T. Kenyon. 2017. Generating sparse representations using quantum annealing: Comparison to classical algorithms. In *Proceedings of the IEEE International Conference on Rebooting Computing (ICRC'17)*. 1–6. DOI: https://doi.org/10.1109/ICRC.2017.8123653
[183] Michael A. Nielsen and Isaac L. Chuang. 2010. *Quantum Computation and Quantum Information*. Cambridge University Press.
[184] M. A. Novotny, Q. L. Hobl, J. S. Hall, and K. Michielsen. 2016. Spanning tree calculations on d-Wave 2 machines. In *Journal of Physics: Conference Series*, Vol. 681. IOP Publishing, 012005.
[185] Thomas R. Osborn. 1990. *Fast Teaching of Boltzmann Machines with Local Inhibition*. Springer Netherlands, Dordrecht, 785–785.
[186] Peter J. J. O'Malley, Ryan Babbush, Ian D. Kivlichan, Jonathan Romero, Jarrod R. McClean, Rami Barends, Julian Kelly, Pedram Roushan, Andrew Tranter, Nan Ding, et al. 2016. Scalable quantum simulation of molecular energies. *Phys. Rev. X* 6, 3 (2016), 031007.
[187] Eustace Painkras, Luis A. Plana, Jim Garside, Steve Temple, Francesco Galluppi, Cameron Patterson, David R. Lester, Andrew D. Brown, and Steve B. Furber. 2013. SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation. *IEEE J. Solid-State Circ.* 48, 8 (2013), 1943–1953.
[188] Scott Pakin and Steven P. Reinhardt. 2018. A survey of programming tools for D-wave quantum-annealing processors. In *Proceedings of the International Conference on High Performance Computing*. Springer, 103–122.
[189] Ojas Parekh, Jeremy Wendt, Luke Shulenburger, Andrew Landahl, Jonathan Moussa, and John Aidun. 2016. Benchmarking adiabatic quantum optimization for complex network analysis. *arXiv preprint arXiv:1604.00319* (2016).
[190] N. G. Pavlidis, O. K. Tasoulis, Vassilis P. Plagianakos, G. Nikiforidis, and M. N. Vrahatis. 2005. Spiking neural network training using evolutionary algorithms. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'05)*, Vol. 4. IEEE, 2190–2194.
[191] Bruno U. Pedroni, Srinjoy Das, John V. Arthur, Paul A. Merolla, Bryan L. Jackson, Dharmendra S. Modha, Kenneth Kreutz-Delgado, and Gert Cauwenberghs. 2016. Mapping generative models onto a network of digital spiking neurons. *IEEE Trans. Biomed. Circ. Syst.* 10, 4 (2016), 837–854.
[192] Alejandro Perdomo-Ortiz, Marcello Benedetti, John Realpe-Gómez, and Rupak Biswas. 2018. Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. *Quant. Sci. Technol.* 3, 3 (2018), 030502.
[193] Alejandro Perdomo-Ortiz, Alexander Feldman, Asier Ozaeta, Sergei V. Isakov, Zheng Zhu, Bryan O'Gorman, Helmut G. Katzgraber, Alexander Diedrich, Hartmut Neven, Johan de Kleer, Brad Lackey, and Rupak Biswas. 2019. Readiness of quantum optimization machines for industrial applications. *Phys. Rev. Appl.* 12 (July 2019), 014004. Issue 1. DOI: https://doi.org/10.1103/PhysRevApplied.12.014004
[194] Yuriy V. Pershin and Massimiliano Di Ventra. 2010. Experimental demonstration of associative memory with memristive neural networks. *Neural Netw.* 23, 7 (2010), 881–886.

[195] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. 2014. A variational eigenvalue solver on a photonic quantum processor. *Nat. Commun.* 5 (2014), 4213.

[196] James S. Plank, Garrett S. Rose, Mark E. Dean, Catherine D. Schuman, and Nathaniel C. Cady. 2017. A unified hardware/software co-design framework for neuromorphic computing devices and applications. In *Proceedings of the IEEE International Conference on Rebooting Computing (ICRC'17)*. IEEE, 1–8.

[197] Filip Jan Ponulak and John J. Hopfield. 2013. Rapid, parallel path planning by propagating wavefronts of spiking neural activity. *Front. Computat. Neurosci.* 7 (2013), 98.

[198] Thomas E. Potok, Catherine D. Schuman, Steven R. Young, Robert M. Patton, Federico Spedalieri, Jeremy Liu, Ke-Thia Yao, Garrett Rose, and Gangotree Chakma. 2016. A study of complex deep learning networks on high performance, neuromorphic, and quantum computers. In *Proceedings of the Workshop on Machine Learning in HPC Environments (MLHPC'16)*. IEEE, 47–55.

[199] John Preskill. 2018. Quantum computing in the NISQ era and beyond. *arXiv preprint arXiv:1801.00862* (2018).

[200] Mirko Prezioso, Farnood Merrikh-Bayat, B. D. Hoskins, G. C. Adam, Konstantin K. Likharev, and Dmitri B. Strukov. 2015. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* 521, 7550 (2015), 61.

[201] PsiQuantum. 2019. PsiQ. Retrieved from http://psiquantum.com/.

[202] Marcos G. Quiles, Liang Zhao, Fabricio A. Breve, and Anderson Rocha. 2010. Label propagation through neuronal synchrony. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'10)*. IEEE, 1–8.

[203] Marcos G. Quiles, Ezequiel R. Zorzal, and Elbert E. N. Macau. 2013. A dynamical model for community detection in complex networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'13)*. IEEE, 1–8.

[204] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. 2009. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th International Conference on Machine Learning*. ACM, 873–880.

[205] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. 2014. Quantum support vector machine for big data classification. *Phys. Rev. Lett.* 113, 13 (2014), 130503.

[206] Jörg Reichardt and Stefan Bornholdt. 2004. Detecting fuzzy community structures in complex networks with a Potts model. *Phys. Rev. Lett.* 93, 21 (2004), 218701.

[207] Jörg Reichardt and Stefan Bornholdt. 2006. Statistical mechanics of community detection. *Phys. Rev. E* 74 (July 2006), 016110. Issue 1. DOI: https://doi.org/10.1103/PhysRevE.74.016110

[208] Eleanor G. Rieffel, Davide Venturelli, Bryan O'Gorman, Minh B. Do, Elicia M. Prystay, and Vadim N. Smelyanskiy. 2015. A case study in programming a quantum annealer for hard operational planning problems. *Quant. Inf. Proc.* 14, 1 (2015), 1–36.

[209] Rigetti. 2018. The Rigetti QPU. Retrieved from http://www.rigetti.com/qpu.

[210] Diego Ristè, Marcus P. da Silva, Colm A. Ryan, Andrew W. Cross, Antonio D. Córcoles, John A. Smolin, Jay M. Gambetta, Jerry M. Chow, and Blake R. Johnson. 2017. Demonstration of quantum advantage in machine learning. *npj Quant. Inf.* 3, 1 (2017), 16.

[211] Peter Ronhovde and Zohar Nussinov. 2010. Local resolution-limit-free Potts model for community detection. *Phys. Rev. E* 81, 4 (2010), 046114.

[212] Frank Rosenblatt. 1961. *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms*. Technical Report. Cornell Aeronautical Lab Inc, Buffalo, NY.

[213] Christopher Rozell, Don Johnson, Richard Baraniuk, and Bruno Olshausen. 2007. Locally competitive algorithms for sparse approximation. In *Proceedings of the IEEE International Conference on Image Processing. (ICIP'07)*, Vol. 4. IEEE, IV–169.

[214] Christopher J. Rozell, Don H. Johnson, Richard G. Baraniuk, and Bruno A. Olshausen. 2008. Sparse coding via thresholding and local competition in neural circuits. *Neural Computat.* 20, 10 (2008), 2526–2563.

[215] Krishna Kumar Sabapathy, Haoyu Qi, Josh Izaac, and Christian Weedbrook. 2019. Production of photonic universal quantum gates enhanced by machine learning. *Phys. Rev. A* 100 (July 2019), 012326. Issue 1. DOI: https://doi.org/10.1103/PhysRevA.100.012326

[216] Siddhartha Santra, Gregory Quiroz, Greg Ver Steeg, and Daniel A. Lidar. 2014. MAX 2-SAT with up to 108 qubits. *New J. Phys.* 16, 4 (2014), 045006.

[217] Siddhartha Santra, Omar Shehab, and Radhakrishnan Balu. 2017. Ising formulation of associative memory models and quantum annealing recall. *Phys. Rev. A* 96, 6 (2017), 062330.

[218] Michael T. Schaub, Jean-Charles Delvenne, Martin Rosvall, and Renaud Lambiotte. 2017. The many facets of community detection in complex networks. *Appl. Netw. Sci.* 2, 1 (2017), 4. DOI: https://doi.org/10.1007/s41109-017-0023-6

[219] Thomas Schiex, Helene Fargier, Gerard Verfaillie, et al. 1995. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*. 631–639.

[220] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Netw.* 61 (2015), 85–117.

[221] Benjamin Schrauwen, Michiel D'Haene, David Verstraeten, and Jan Van Campenhout. 2008. Compact hardware liquid state machines on FPGA for real-time speech recognition. *Neural Netw.* 21, 2 (2008), 511–523.

[222] Jonathan Schrock, Alex J. McCaskey, Kathleen E. Hamilton, Travis S. Humble, and Neena Imam. 2017. Recall performance for content-addressable memory using adiabatic quantum optimization. *Entropy* 19, 9 (2017), 500.

[223] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. 2019. Evaluating analytic gradients on quantum hardware. *Phys. Rev. A* 99, 3 (2019), 032331.

[224] Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. 2018. Circuit-centric quantum classifiers. *arXiv preprint arXiv:1804.00633* (2018).

[225] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. 2014. The quest for a quantum neural network. *Quant. Inf. Proc.* 13, 11 (2014), 2567–2586.

[226] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. 2015. An introduction to quantum machine learning. *Contemp. Phys.* 56, 2 (2015), 172–185.

[227] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. 2015. Simulating a perceptron on a quantum computer. *Phys. Lett. A* 379, 7 (2015), 660–663.

[228] Catherine D. Schuman, Kathleen Hamilton, Tiffany Mintz, Md Musabbir Adnan, Bon Woong Ku, Sung-Kyu Lim, and Garrett S. Rose. 2019. Shortest path and neighborhood subgraph extraction on a spiking memristive neuromorphic implementation. In *Proceedings of the Neuro-Inspired Computational Elements (NICE'19) Workshop.*

[229] Catherine D. Schuman, James S. Plank, Adam Disney, and John Reynolds. 2016. An evolutionary optimization framework for neural networks and neuromorphic architectures. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'16).* IEEE, 145–154.

[230] Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, and James S. Plank. 2017. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963* (2017).

[231] Hadayat Seddiqi and Travis S. Humble. 2014. Adiabatic quantum optimization for associative memory recall. *Front. Phys.* 2 (2014), 79.

[232] Jae-sun Seo, Bernard Brezzo, Yong Liu, Benjamin D. Parker, Steven K. Esser, Robert K. Montoye, Bipin Rajendran, José A. Tierno, Leland Chang, Dharmendra S. Modha, et al. 2011. A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC'11).* IEEE, 1–4.

[233] William Severa, Craig M. Vineyard, Ryan Dellana, Stephen J. Verzi, and James B. Aimone. 2019. Training deep neural networks for binary communication with the Whetstone method. *Nat. Mach. Intell.* 1, 2 (2019), 86.

[234] William M. Severa, Craig M. Vineyard, Ryan Dellana, and James B. Aimone. 2018. Whetstone: An accessible, platform-independent method for training spiking deep neural networks for neuromorphic processors. In *Proceedings of the Neuro Inspired Computational Elements Workshop.*

[235] Jeffrey M. Shainline, Sonia M. Buckley, Richard P. Mirin, and Sae Woo Nam. 2017. Superconducting optoelectronic circuits for neuromorphic computing. *Phys. Rev. Appl.* 7, 3 (2017), 034013.

[236] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Ilya Safro, Susan Mniszewski, and Yuri Alexeev. 2018. Community detection across emerging quantum architectures. *arXiv preprint arXiv:1810.07765* (2018).

[237] David Sherrington and Scott Kirkpatrick. 1975. Solvable model of a spin-glass. *Phys. Rev. Lett.* 35, 26 (1975), 1792.

[238] Taime Shoji, Kazuyuki Aihara, and Yoshihisa Yamamoto. 2017. Quantum model for coherent Ising machines: Stochastic differential equations with replicator dynamics. *Phys. Rev. A* 96, 5 (Nov. 2017). DOI: https://doi.org/10.1103/physreva.96.053833

[239] Peter W. Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* 41, 2 (1999), 303–332.

[240] Sumit Bam Shrestha and Garrick Orchard. 2018. SLAYER: Spike layer error reassignment in time. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems.* 1412–1421.

[241] Hava T. Siegelmann. 2003. Neural and super-turing computing. *Minds Mach.* 13, 1 (2003), 103–114.

[242] Haozhen Situ, Zhimin He, Lvzhou Li, and Shenggen Zheng. 2018. Quantum generative adversarial network for generating discrete data. *arXiv preprint arXiv:1807.01235* (2018).

[243] Robert S. Smith, Michael J. Curtis, and William J. Zeng. 2016. A practical quantum instruction set architecture. *arXiv preprint arXiv:1608.03355v2* (2017).

[244] Friedrich T. Sommer and Thomas Wennekers. 2001. Associative memory in networks of spiking neurons. *Neural Netw.* 14, 6–7 (2001), 825–834.

[245] Fengguang Song, Stanimire Tomov, and Jack Dongarra. 2012. Enabling and scaling matrix computations on heterogeneous multi-core and multi-GPU systems. In *Proceedings of the 26th ACM International Conference on Supercomputing.* ACM, 365–376.

[246] Sen Song, Kenneth D. Miller, and Larry F. Abbott. 2000. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci.* 3, 9 (2000), 919.

[247] Amos J. Storkey and Romain Valabregue. 1999. The basins of attraction of a new Hopfield learning rule. *Neural Netw.* 12, 6 (1999), 869–876.

[248] Evangelos Stromatias, Daniel Neil, Francesco Galluppi, Michael Pfeiffer, Shih-Chii Liu, and Steve Furber. 2015. Scalable energy-efficient, low-latency implementations of trained spiking deep belief networks on SpiNNaker. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'15)*. IEEE, 1–8.

[249] Francesco Tacchino, Chiara Macchiavello, Dario Gerace, and Daniele Bajoni. 2018. An artificial neuron implemented on an actual quantum processor. *arXiv preprint arXiv:1811.02266* (2018).

[250] Kenta Takata, Alireza Marandi, and Yoshihisa Yamamoto. 2015. Quantum correlation in degenerate optical parametric oscillators with mutual injections. *Phys. Rev. A* 92, 4 (Oct. 2015). DOI: https://doi.org/10.1103/physreva.92.043821

[251] Y. Takeda, S. Tamate, Y. Yamamoto, H. Takesue, T. Inagaki, and S. Utsunomiya. 2017. Boltzmann sampling for an XY model using a non-degenerate optical parametric oscillator network. *Quant. Sci. Technol.* 3, 1 (Nov. 2017), 014004. DOI: https://doi.org/10.1088/2058-9565/aa923b

[252] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. 2017. Error mitigation for short-depth quantum circuits. *Phys. Rev. Lett.* 119, 18 (2017), 180509.

[253] Hayato Ushijima-Mwesigwa, Christian F. A. Negre, and Susan M. Mniszewski. 2017. Graph partitioning using quantum annealing on the D-wave system. *arXiv preprint arXiv:1705.03082* (2017).

[254] Sacha Jennifer van Albada, Andrew G. Rowley, Johanna Senk, Michael Hopkins, Maximilian Schmidt, Alan Barry Stokes, David R. Lester, Markus Diesmann, and Steve B. Furber. 2018. Performance comparison of the digital neuromorphic hardware SpiNNaker and the neural network simulation software NEST for a full-scale cortical microcircuit model. *Front. Neurosci.* 12 (2018), 291.

[255] Peter J. M. Van Laarhoven and Emile H. L. Aarts. 1987. Simulated annealing. In *Simulated Annealing: Theory and Applications*. Springer, 7–15.

[256] Davide Venturelli, Salvatore Mandrà, Sergey Knysh, Bryan O'Gorman, Rupak Biswas, and Vadim Smelyanskiy. 2015. Quantum optimization of fully connected spin glasses. *Phys. Rev. X* 5, 3 (2015), 031040.

[257] Guillaume Verdon, Juan Miguel Arrazola, Kamil Brádler, and Nathan Killoran. 2019. A quantum approximate optimization algorithm for continuous problems. *arXiv preprint arXiv:1902.00409* (2019).

[258] Guillaume Verdon, Michael Broughton, and Jacob Biamonte. 2017. A quantum algorithm to train neural networks using low-depth circuits. *arXiv preprint arXiv:1712.05304* (2017).

[259] Guillaume Verdon, Jason Pye, and Michael Broughton. 2018. A universal training algorithm for quantum deep learning. *arXiv preprint arXiv:1806.09729* (2018).

[260] Jeffrey S. Vetter, Ron Brightwell, Maya Gokhale, Pat McCormick, Rob Ross, John Shalf, Katie Antypas, David Donofrio, Travis Humble, Catherine Schuman, et al. 2019. *Extreme Heterogeneity 2018-Productive Computational Science in the Era of Extreme Heterogeneity: Report for DOE ASCR Workshop on Extreme Heterogeneity*. Technical Report. Department of Energy, Office of Science.

[261] Jeffrey S. Vetter, Richard Glassbrook, Jack Dongarra, Karsten Schwan, Bruce Loftis, Stephen McNally, Jeremy Meredith, James Rogers, Philip Roth, Kyle Spafford, et al. 2011. Keeneland: Bringing heterogeneous GPU computing to the computational science community. *Comput. Sci. Eng.* 13, 5 (2011), 90–95.

[262] Leyuan Wang, Yangzihao Wang, Carl Yang, and John D. Owens. 2016. A comparative study on exact triangle counting algorithms on the GPU. In *Proceedings of the ACM Workshop on High Performance Graph Processing*. ACM, 1–8.

[263] Zhe Wang, Alireza Marandi, Kai Wen, Robert L. Byer, and Yoshihisa Yamamoto. 2013. Coherent Ising machine based on degenerate optical parametric oscillators. *Phys. Rev. A* 88, 6 (Dec. 2013). DOI: https://doi.org/10.1103/physreva.88.063853

[264] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. 2016. Training a quantum optimizer. *Phys. Rev. A* 94, 2 (2016), 022309.

[265] Arman Zaribafiyan, Dominic J. J. Marchand, and Seyed Saeed Changiz Rezaei. 2017. Systematic and deterministic graph minor embedding for Cartesian products of graphs. *Quant. Inf. Proc.* 16, 5 (2017), 136.

[266] Jinfeng Zeng, Yufeng Wu, Jin-Guo Liu, Lei Wang, and Jiangping Hu. 2019. Learning and inference on generative adversarial quantum circuits. *Phys. Rev. A* 99, 5 (2019), 052306.

[267] Fang Zhang, Cupjin Huang, Michael Newman, Junjie Cai, Huanjun Yu, Zhengxiong Tian, Bo Yuan, Haihong Xu, Junyin Wu, Xun Gao, Jianxin Chen, Mario Szegedy, and Yaoyun Shi. 2019. Alibaba cloud quantum development kit: Large-scale classical simulation of quantum circuits. *arXiv e-prints* arXiv:1907.11217 (July 2019).

[268] Kenneth M. Zick, Omar Shehab, and Matthew French. 2015. Experimental quantum annealing: Case study involving the graph isomorphism problem. *Sci. Rep.* 5 (2015), 11168.

Q9 Received XXXX; revised XXXX; accepted XXXX

Author Queries

Q1: AU: "...a universal approximator if nonlinear functions." written as meant?
Q2: AU: "instruction follows" written as meant?
Q3: AU: References 29 and 30 are identical.
Q4: AU: Please supply volume and issue numbers for Reference 65.
Q5: AU: Please supply year of publication for Reference 88.
Q6: AU: Please supply year of publication for Reference 134.
Q7: AU: Please supply volume and issue numbers for Reference 154.
Q8: AU: Please supply volume and issue numbers for Reference 176.
Q9: AU: Please provide article history date.