Quantum Science and Technology

OPEN ACCESS

RECEIVED
23 October 2024

REVISED
27 February 2025

ACCEPTED FOR PUBLICATION
21 March 2025

PUBLISHED
28 March 2025

PAPER

A memristive neural decoder for cryogenic fault-tolerant quantum error correction

Victor Yon[1,2,3,4,9,*], Frédéric Marcotte[2,3,4,5,9], Pierre-Antoine Mouny[1,2,3,4], Gebremedhin A Dagnew[5], Bohdan Kulchytskyy[5], Sophie Rochette[1], Yann Beilliard[2,3,4,5], Dominique Drouin[1,2,3,4] and Pooya Ronagh[1,6,7,8,*]

1 Irréversible Inc., Sherbrooke, Québec, Canada
2 Institut Interdisciplinaire d'Innovation Technologique (3IT), Université de Sherbrooke, Sherbrooke, Québec, Canada
3 Laboratoire Nanotechnologies Nanosystèmes (LN2 - IRL 3463) – CNRS, Université de Sherbrooke, Sherbrooke, Québec, Canada
4 Institut quantique (IQ), Université de Sherbrooke, Sherbrooke, Québec, Canada
5 1QB Information Technologies (1QBit), Vancouver, British Columbia, Canada
6 Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario, Canada
7 Department of Physics & Astronomy, University of Waterloo, Waterloo, Ontario, Canada
8 Perimeter Institute for Theoretical Physics, Waterloo, Ontario, Canada
9 These authors contributed equally to this work.
* Authors to whom any correspondence should be addressed.

E-mail: victor.yon@usherbrooke.ca and pooya@irreversible.tech

Keywords: quantum error correction, surface code, memristor, artificial neural network, quantum error decoder, in-memory computing

Supplementary material for this article is available online

Abstract
Neural decoders for quantum error correction rely on neural networks to classify syndromes extracted from error correction codes and find appropriate recovery operators to protect logical information against errors. Its ability to adapt to hardware noise and long-term drifts make neural decoders promising candidates for inclusion in a fault-tolerant quantum architecture. However, given their limited scalability, it is prudent that small-scale (local) neural decoders are treated as first stages of multi-stage decoding schemes for fault-tolerant quantum computers with millions of qubits. In this case, minimizing the decoding time to match the stabilization measurements frequency and a tight co-integration with the QPUs is highly desired. Cryogenic realizations of neural decoders can not only improve the performance of higher stage decoders, but they can minimize communication delays, and alleviate wiring bottlenecks. In this work, we design and analyze a neural decoder based on an in-memory computation (IMC) architecture, where crossbar arrays of resistive memory devices are employed to both store the synaptic weights of the neural decoder and perform analog matrix–vector multiplications. In simulations supported by experimental measurements, we investigate the impact of TiO_x-based memristive devices' non-idealities on decoding fidelity. We develop hardware-aware re-training methods to mitigate the fidelity loss, restoring the ideal decoder's pseudo-threshold for the distance-3 surface code. This work provides a pathway to scalable, fast, and low-power cryogenic IMC hardware for integrated fault-tolerant quantum error correction.

1. Introduction

Fault-tolerant quantum computation (FTQC) holds the promise of solving extremely difficult problems with efficient time and space complexity [1, 2]. However, this efficiency is at the cost of resource-intensive classical procedures that are required for protecting the logical quantum state of the quantum processor against noise [3]. This includes, *(a)* physically protecting the quantum state by cooling and isolating the quantum processor, and *(b)* performing quantum control and quantum error correction (QEC) protocols. The former is the reason many quantum technologies operate at cryogenic temperatures. But, the latter requires classical computing circuits that have historically been designed and manufactured to operate at room temperature and their high heat dissipations hinders their use inside cryostats.

The transfer and processing of the data generated by QEC protocols present numerous challenges: *(i)* With the anticipated tens of millions of physical qubits needed for FTQC, the repeated microsecond-long error correction cycles would produce terabytes of syndrome data per second to be processed for days- or even months-long computations [2, 4, 5]. Transferring this amount of data from the cryogenic environment to classical electronics located at room temperature quickly leads to wiring bottlenecks [6]. *(ii)* Transferring small signals from measurement at the quantum processor level through multiple temperature stages to be processed by room temperature instrumentation requires careful amplification to mitigate the increased thermal noise at each stage. Finally, *(iii)*, an FTQC implementation of non-Clifford gates requires active error correction, that is, real-time processing of the syndrome data by a decoder module, and application of the resulting recovery operations at a time scale comparable to the coherence time of the quantum system [7, 8]. Classical processors capable of operating accurately at cryogenic temperatures can mitigate these challenges. This requires computing with extremely low heat dissipation, as the cooling power of dilution refrigerators is limited (1 to 2 W at the 4 K stage).

Since single-flux (SFQ) quantum digital electronics are natively compatible with cryogenic temperatures, recent works [9–11] propose SFQ-based architectures for decoding using various heuristic approximations to graph-matching algorithms [12–14], while [11] proposes a binarized SFQ-based neural decoders. Those systems are highly complex and consume a lot of energy, casting doubts on the ability to fabricate them reliably and on their potential benefits. The main challenge in building scalable cryogenic fault-tolerant decoders is the need for high-density cryogenic memory blocks for storing at least two types of information:

(a) The program data: Even the simplest decoding heuristics require complex computations that rely on large look-up tables to store the logic and execute the algorithmic steps of the decoder [15, 16].
(b) The input data: Fault-tolerant error correction requires processing multiple rounds of imperfect stabilizer measurements (typically as a graph-based algorithm on a 3-dimensional lattice) at once. Moreover, current decoding logic depends on historical progression of the algorithm [17–19]. Therefore, some type of memory must store and recall a historical state of computation.

In this paper, we introduce a memristive neural decoder (MND) which combines the advantages of recurrent neural networks (RNNs) with the low-power consumption of resistive memory arrays to eliminate the issues related to storage and recall for both types of memory blocks. In neural decoders, the program data (item *(i)* above) is the weights and biases of a neural network [20]. This information is stored in the conductance states of the resistive memory devices which are physically tuned, thereby providing a realization of in-memory computation (IMC) [21]. In addition, an RNN comprises an *internal state* which is a real-valued vector (or a higher-dimensional tensor) that encodes a latent representation of its prior inference (the h wires in figure 1). The RNN will therefore not need to receive the syndrome data of multiple rounds of error correction at once, and instead stream their processing one at a time as they are generated, while updating its internal state in each iteration (from h_i to h_{i+1} in figure 1).

More specifically, we investigate TiO_x-based analog resistive memory devices using TiN electrodes [22]. A crossbar arrangement of these memory cells enables the matrix–vector multiplication (MVM) operations at the heart of neural network algorithms to be performed natively by relying on Ohm's law and Kirchoff's circuit law, thus removing the time- and energy-intensive process of moving data from memory to processing units [23]. Such non-volatile memristive devices have small footprints [24–27], and benefit from complementary metal-oxide-semiconductor (CMOS)-compatible fabrication processes [22], data retention time of up to 10 years [28], and analog switching dynamics [29], making them promising candidates for efficient MVM in terms of processing time, energy dissipation, and scalability [30–32]. Furthermore, they operate very well at cryogenic temperatures [33–35], are robust to temperature variations, and can be calibrated to adjust to long-term drifts in the input signal or environmental noise.

Moreover, the MND processes neural activations (the feature tensors propagating forward along the network) as inherently analog signals which are never converted to digital data in binary representation. This allows for a much better footprint compared to the alternative CMOS-based digital (binary) memory technologies such as static random access memory (SRAM) and resistive random access memory which were previously considered for low-power digital neural decoding [36, 37]. We note that the mixed-signal nature of MND makes it attractive for soft decoding [38] by directly interfacing the readout resonator and eliminating long and error-prone amplification and measurement of syndrome bits. Since it is difficult to scale neural decoders up for large surface code patches, we envision that such a tightly integrated decoder–quantum processing unit hybrid system may be augmented with further (more global) decoding stages such as large scale Union–Find or collision clustering decoders [39, 40].

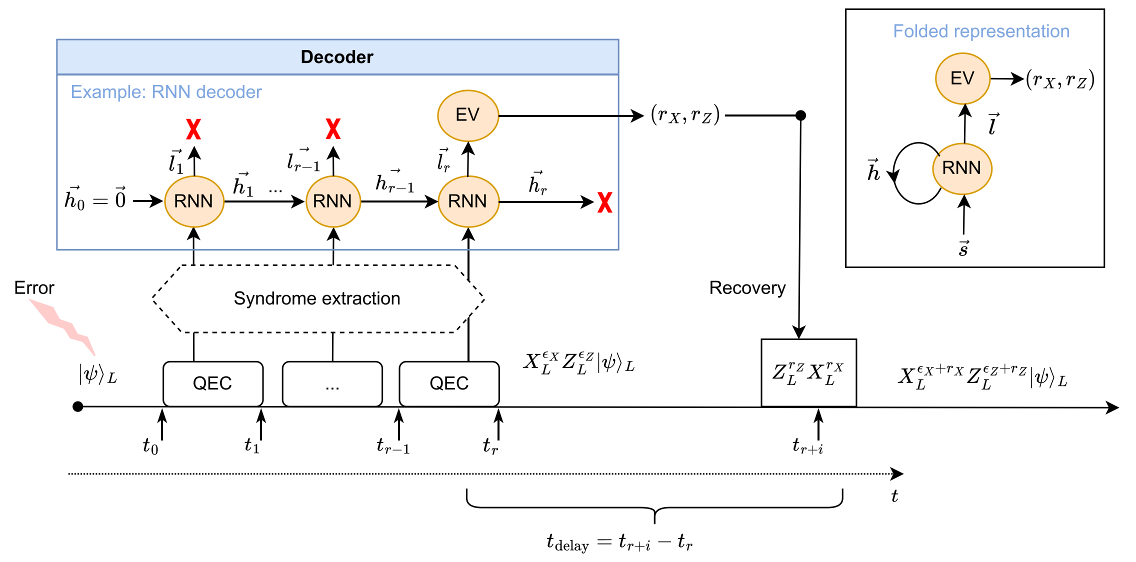

Figure 1. Decoding process. In quantum error correction (QEC), syndrome extraction rounds are performed cyclically to protect an encoded quantum state $|\psi\rangle_L$ from logical errors. Typically, the X and Z stabilizers' syndromes are processed separately but simultaneously by two independent decoding modules. The syndrome measurement outcomes are fed to the decoder modules, which analyze them to produce a decision output (r_X, r_Z). This decision output is used to construct the recovery operation $Z_L^{r_Z} X_L^{r_X}$. Between the end of the last QEC round at t_r and the application of the recovery operation at t_{r+i}, idling errors can affect the qubits, and they will not be taken into account by the decoding algorithm. The encoded state just prior to the application of recovery operations is therefore $X_L^{\epsilon_X} Z_L^{\epsilon_Z} |\psi\rangle_L$, where $\epsilon_{x,z}$ represents the cumulative effect of the errors that occur during the QEC rounds and the idling time t_{delay}. Following the application of the recovery operation (here assumed to be ideal for simplicity), the final encoded logical quantum state is $X_L^{\epsilon_X + r_X} Z_L^{\epsilon_Z + r_Z} |\psi\rangle_L$. Here, the example decoder is based on a recurrent neural network (RNN). Each new syndrome is used as the input to the RNN and the hidden state $\vec{h}_i$ is passed to the subsequent recurrence. At the end of the QEC process, the output state $\vec{l}_r$ is passed to an evaluation module evaluation module (EV) (the fully connected output layer in the case of an RNN) to provide the final binary output of the decoder. On the right-hand side of the figure, the standard folded representation of this RNN is illustrated. A more detailed schematic architecture of this RNN decoder is depicted in figure 3.

Our early MND prototype is fabricated in CMOS 180 nm and demonstrates cryogenic compatibility down to 35 K [41]. It exhibits a decoding delay of 1 μs per stabilizer measurement round, which is similar to Collision Clustering decoders able to perform real-time QEC by demonstrating a 2×2 stability experiment [42]. Our decoder delay is currently bottlenecked by the pulse width used at cryogenic temperature and could be reduced to 200 ns with minimal CMOS design optimization. It is expected that our processing delay will not increase significantly with larger QEC codes as analog RNN allows for parallel computation. Additionally, the MND prototype consumes 3.4 mW at 35 K to perform the RNN computation and 10.9 mW for the auxiliary electronics used to interface with the memristors. It is anticipated that the distance-3 MND will consume $\sim$50 mW for the RNN computation and roughly 120 mW for the memristor/CMOS interfacing electronics in CMOS 180 nm. The power consumption can be significantly reduced by using smaller CMOS nodes, e.g. 22 nm FDSOI exhibits power consumption up to 70 times smaller than CMOS 180 nm [43], yielding to a 2.5 mW power consumption per distance-3 MND.

However, non-idealities of memrsitor arrays are known to deteriorate neural networks (NNs)' performance [44–48], our study is focused on the impact of key TiO$_x$-based devices' non-idealities on the accuracy of MNDs. Stuck-at fault devices have the greatest impact on the decoder's performance. Therefore, we propose and implement two techniques for mitigating their detrimental effect. The hardware-aware (HWA) method consists of improving NNs' robustness by re-training it while taking into account typical hardware constraints of memristors. In contract, the device-specific (DS) re-training method uses the exact location of stuck-at fault devices to specifically adapt to the imperfections of a given device. We show that the latter approach allows for high-fidelity decoding of the distance-3 surface code.

Our paper is structured as follows. First, we provide a formal definition of the decoding problem using a distance-3 surface code, we describe the NN architecture of our decoder, and present the corresponding memristive decoder circuit. We then experimentally characterize key non-idealities of the TiO$_x$-based resistive memory devices (e.g. programming variability, stuck-at fault rate, retention time). We then assess the impact of these non-idealities in simulations, and demonstrate that the HWA and (DS) re-training methods recover most of the ideal neural decoder's fidelity. Finally, we discuss the engineering advantages of leveraging analog IMC hardware for QEC decoding, and provide some future perspectives.

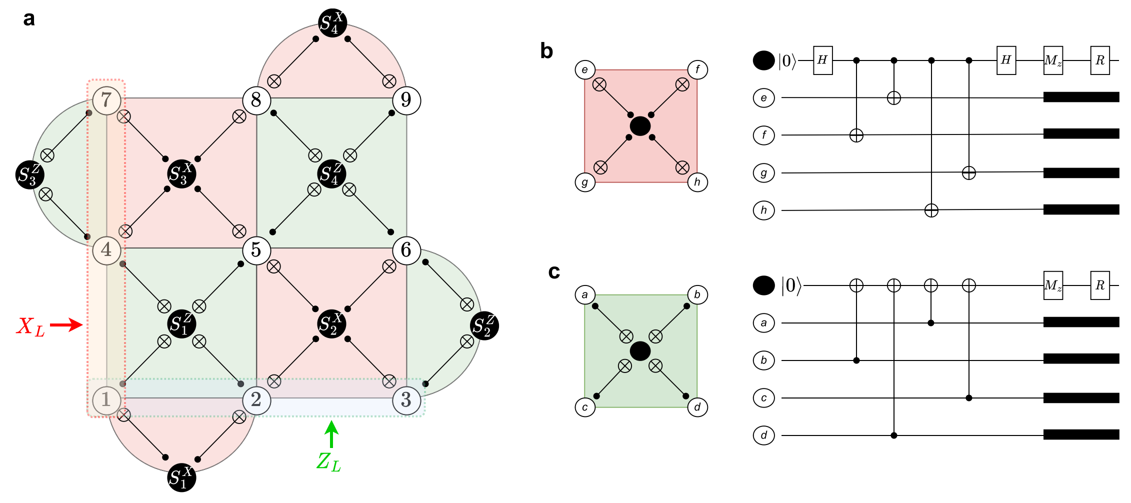

Figure 2. Surface code and stabilizer measurements. (a) Distance-3 rotated surface code, also called surface-17. Data qubits 1 to 9 are identified by circles filled in white, and syndrome qubits are identified by circles filled in black. Each syndrome qubit resides on a 'plaquette' corresponding to a specific stabilizer measurement S. The sequence of operations realizing an X stabilizer (pink plaquette) and a Z stabilizer (green plaquette) are shown in (b) and (c). In this example, a logical operator X_L (Z_L) can be realized by applying a chain of physical X (Z) operators on data qubits between the top (left) and bottom (right) edge of the grid. (b) Sequence of circuit operations between a syndrome qubit (circle filled in black, the top qubit in the circuit) and its neighboring data qubits, e, f, g, and h, realizing the stabilizer measurement $X_eX_fX_gX_h$. It consists of syndrome qubit initialization, Hadamard gates (H), CNOT gates, a projective measurement along the z-axis (M_Z), and a reset (R) to the $|0\rangle$ state, regardless of the measured outcome. (c) Sequence of circuit operations between a syndrome qubit and its neighboring data qubits, a, b, c, and d, realizing the stabilizer measurement $Z_aZ_bZ_cZ_d$. It consists of syndrome qubit initialization, CNOT gates, a projective measurement along the z-axis (M_Z), and a reset (R) to the $|0\rangle$ state, regardless of the measured outcome. The black rectangles represent idling of the data qubits, which is important for simulation of the circuits. More details on the exact procedure used for the simulation of these parity-check circuits is provided in supplementary information note S1.

2. Problem statement

2.1. Decoding the surface code

In figure 1, the process of active error correction via stabilizer measurements and decoding is schematized. Here, successful error correction amounts to matching the decoder-proposed recovery operator (r_X, r_Z) with the logical error (ϵ_X, ϵ_Z) afflicting the logical state $|\psi\rangle_L$ encoded by the error correcting code. The performance of the decoder is dependent on its speed due to idling errors associated with the decoding delay. Therefore, minimizing the decoding time as much as possible is desirable.

We consider the distance-3 rotated surface code, which can be realized with 17 physical qubits [49–51]. As shown in figure 2(a), the qubits are arranged on a square lattice, comprising data qubits (circles filled in white) and ancilla, or syndrome, qubits (circles filled in black). Data and syndrome qubits differ only in terms of their function within the code, and they can be implemented using physical systems such as superconducting circuits, trapped ions, quantum dots, or topological qubits. Each qubit interacts with its neighbors in a specified manner. The order and mechanism of interaction is determined by the stabilizers being measured, as shown in figures 2(b) and (c) for the stabilizers $X_eX_fX_gX_h$ and $Z_aZ_bZ_cZ_d$, respectively.

2.2. Neural decoder architecture

We consider an RNN decoder module similar to the ones described in [20] and [52]. It may be difficult to train neural decoders for arbitrarily large topological codes; however, we note that the largest topological patch that must be actively decoded during FTQC depends on the largest entangling gate between a logical magic state and other logical qubits [53]. Additionally, neural decoders are great candidates for inclusion in distributed [54] and hierarchical decoding schemes [55] in order to create larger-scale decoding systems. We restrict our benchmarking to the X syndromes because the performance would be the same for the Z syndromes. supplementary information note S1 describes the model used to simulate the quantum circuit and obtain the syndromes datasets [56] and labels for training the RNN decoder.

Our RNN architecture is illustrated in figure 3(b). It consists of a fully connected recurrent layer and a fully connected output evaluation layer. There are 4 input nodes for receiving the syndrome data of an error correction round and 32 internal state nodes to store the hidden state of the previous round via recurrent connections. Assuming similar error rates for physical gates and measurements, the distance-3 code requires three QEC cycles to be fault tolerant [20, 52, 57–60]. The outputs of the recurrent layer, after application of the activation function, a rectified linear unit (ReLU) function in this case, are fed back to the input of the

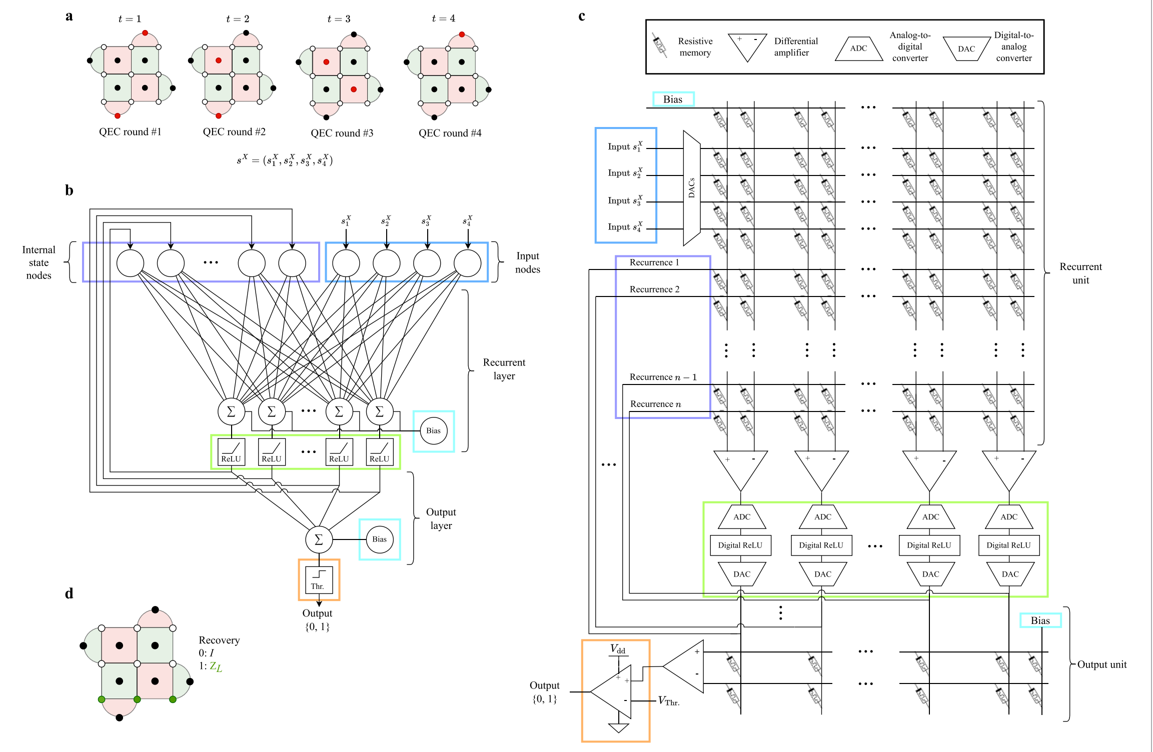

Figure 3. Recurrent neural network decoder architecture and corresponding memristive decoder circuit. Quantum error correction (QEC) is handled as a sequential classification problem. (a) Error correction rounds performed by measuring stabilizer operators on the lattice of physical qubits. Ancilla qubits (the circles filled in red or black representing an error or no error being detected, respectively) measure the bit-flip X (phase-flip Z) stabilizers $\hat{S}^X_{\{1-4\}}$ ($\hat{S}^Z_{\{1-4\}}$). Measured eigenvalues of ancillas are mapped to classical bits to build the syndromes s^X and s^Z, which serve as inputs to the neural decoder. Here, we consider only the s^X syndromes, obtained at times $t = 1, 2, 3, 4$. (b) Recurrent neural network decoder architecture for the distance-3 surface code. It comprises a recurrent layer and an output layer. Bias is also applied to both layers. The syndrome at $t = 1$ is the initial input along with the initialized hidden state $\vec{h}_0$. The output of the recurrent layer is routed back to be used as new input to the internal state nodes, as a new syndrome($t = 2$) is provided to the input nodes. This is repeated until the last QEC round at time $t = n$ (here, $n = 4$), when the output of the recurrent layer is forwarded to the output layer, where the value of the single output neuron is compared to a threshold value to produce a binary classification. (c) Memristive decoder circuit. It is composed of two arrays of memristors. The first is applied recursively (recurrent array) until $t = n$, and the second acts as the classifier (output array). Input syndromes are converted from the digital domain to the analog domain by digital-to-analog converters (DACs) so that the memristive crossbar can perform the analog matrix–vector multiplication. The memristor conductances are programmed according to the digitally trained weights of the equivalent neural network decoder. Differential amplifiers subtract the outputs from each pair of memristor columns to obtain the resulting node's value. This signal is forwarded to the activation function array (green rectangle), where an analog-to-digital conversion is performed to apply a digital ReLU activation function. The signal is converted back to the analog domain before being routed back to the recurrence input ports or sent to the output array. Finally, the value of the output array is passed through a comparator to produce a binary classification. An additional memristor row in each array allows the application of a bias signal. (d) Recovery operation. To conclude the QEC process, a recovery operation is applied to the data qubits according to the classification provided by the decoder. The identity is applied in the case where no logical error has been detected (output 0), and the Z_L operator is applied (circles filled in green) if a logical error has been detected (output 1).

internal state nodes when the next error correction round's syndrome data (s^X) arrives to the input nodes, and the process is repeated for a total of at least 3 cycles, as illustrated in figure 3(a). In order to evaluate the logical error rate of the scheme, we measure the data qubits in the final cycle (see supplementary information note S1).

After the fourth round has been provided to the input nodes, the output of the recurrent layer is forwarded to the output layer and passed through a threshold function. Subsequently, the neural decoder outputs a single binary result, 0 or 1, indicating whether a logical error has occurred at the end of the QEC rounds (see figure 3(d)).

2.3. Memristive neural decoder

We present a memristive electronic circuit to implement the neural decoder architecture discussed above, where the parameters of the neural network are stored in crossbar arrays of TiO_x resistive memory. The MND architecture is shown in figure 3(c). It comprises two distinct memristor arrays: the recurrent array, which maps the weights corresponding to the recurrent layer, and the output array, which maps the weights corresponding to the output layer [61], in accordance with the architecture shown in figure 3(b). Input ports receive data from syndrome measurements in the form of voltage pulses. Between the two crossbars of memristive devices, we perform an analog-to-digital conversion to apply a digital ReLU activation function. The output is then converted back into an analog signal in the form of a voltage pulse. Performing the activation function in the digital domain simplifies the circuit design by removing concerns about analog signal deterioration, as it offers full control over the shape of the signal sent into the second layer. The range and resolution of analog-to-digital converter (ADC) and DAC is discussed in supplementary information note S2. Finally, the output port provides the binary classification result of the decoder, which determines the recovery operation to be applied to the surface code.

The memristor crossbar arrays needed to perform analog MVM consist of two TiN electrodes separated by a TiO_x-based switching layer [22]. Following an initial non-reversible electroforming process, a conductive filament containing oxygen vacancies is created within the oxide layer [62], which can be subsequently at least partially dissolved and re-established through voltage pulses applied on the electrodes, leading to programmable, non-volatile conductance states for the memristive device [63].

It is therefore possible to map the weight matrix of a NN layer into the conductance states $G_{jk}^{\pm}$ of the memristors inside a crossbar array. To implement both positive and negative weights, a differential pair of memristive devices is used (see figure 3(b)). The weight-to-conductance mapping procedure is given by

$$G_{jk}^{\pm} = |w_{jk}| \frac{G_{\mathrm{HCS}} - G_{\mathrm{LCS}}}{w_{\max}} + G_{\mathrm{LCS}}, \tag{1}$$

where $w_{\max}$ is the absolute maximum weight of a given layer, and G_{HCS} and G_{LCS} are the highest conductance states (HCSs) and lowest conductance states (LCSs), respectively. In other words, they are the maximum and minimum values that can be programmed on TiO_x-based memristive devices. If $w_{jk} > 0$, G_{jk}^{+} is programmed with respect to equation (1) and G_{jk}^{-} is set to G_{LCS}. If $w_{jk} < 0$, G_{jk}^{-} is programmed with respect to equation (1) and G_{jk}^{+} is set to G_{LCS}. After the RNN training has completed, the weights can be mapped to conductance states.

A crossbar configuration allows memristive devices to realize MVM natively, by relying on Ohm's law and Kirchhoff's current law. In figure 3(c), the current output of each column in the crossbar array is the sum of the input rows' voltages multiplied by the effective conductance values of the differential pairs of memristors. From the circuit laws, we have

$$i_k = \sum_j \left(G_{jk}^{+} - G_{jk}^{-} \right) v_j, \tag{2}$$

where i and v denote output current and input voltage, respectively. For each differential amplifier, the symbols '$+$' and '$-$' set in superscript form denote the polarity of the pins wired to the memristors. From equation (2), each column implements naturally the multiply-accumulate (MAC) operation [64].

3. Electrical characterization

In this section, we describe and experimentally characterize various hardware non-idealities of TiO_x-based resistive memory devices. Due to the variability of the fabrication process and switching mechanisms, these devices exhibit multiple non-idealities [45, 48, 65, 66], such as read variability, programming variability, and stuck-at fault malfunction (that is, the memory devices become stuck in either HCS or LCS after electroforming or shortly after a conductance programming attempt [67]). These non-idealities are expected to decrease the fidelity of an MND. Other non-idealities such as random telegraphic noise and $1/f$ noise are also commonly reported for oxide-based devices [48]; however, we do not to investigate their impact, as it is expected to be insignificant for high-speed MNDs operating with input voltage pulses in the nanosecond range. Regarding conductance state retention, supplementary information figure S3 shows no noticeable change in the conductance state after 8 hours, confirming the stability of the memory state of these devices, acceptable for target applications. Furthermore, we have recently demonstrated data retention at 4.2 K for over 15 min [34], suggesting that TiO_x-based resistive memory devices are good candidates for a cryogenic MND.

Programming variability, also called cycle-to-cycle variability, is responsible for the inaccurate mapping of trained weights to the conductance states of memristors. Figure 4(a) shows the programming procedure of 11 conductance states on our fabricated memristors using a closed-loop read–write–verify algorithm [63] (see supplementary information note S3). It can be seen that the target values given by equation (1) cannot be reached exactly due to the stochastic nature of the resistive switching process. Therefore, a programming variability model that accounts for cycle-to-cycle variability and device-to-device variability has been obtained from experimental characterizations of TiO_x-based resistive memory devices [34]. The

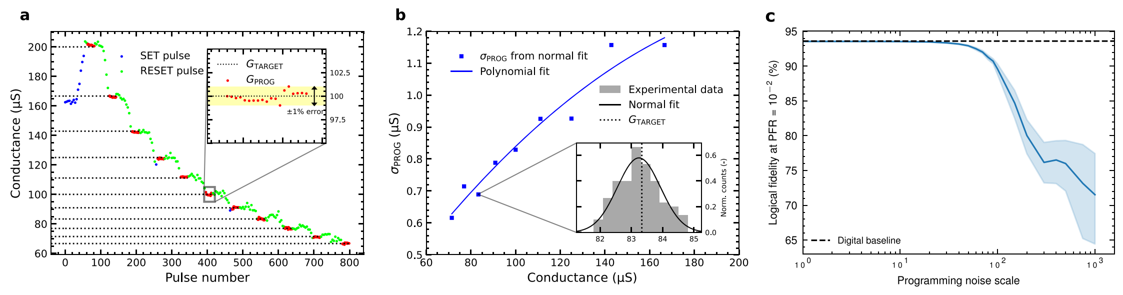

Figure 4. Programming variability characterizations of TiO$_x$-based resistive memory and its impact on the neural decoder. (a) Pulse programming of 11 conductance states of a TiO$_x$-based resistive memory device. Positive voltage pulses induce a conductance increase (labeled 'SET pulse') due to the growth of the conductive filament inside the TiO$_x$ layer between the electrodes. Negative voltage pulses are employed to decrease the conductance ('RESET pulse') due to the rupture of the conductive filament. The dotted black lines denote the target conductance ('G_{TARGET}'), whereas the red dots denote the programmed conductance ('G_{PROG}') values upon convergence of the closed-loop read–write–verify algorithm. The inset is a zoom-in on the readout of the 100 μS state within the allowed error of 1%. (b) Programming variability model based on the programming standard deviations of multiple conductance states. The multilevel programming cycle shown in (a) is conducted 10 times in a double sweep for 10 devices. For each conductance state, the standard deviation of the distribution of programmed values is extracted to fit a programming variability model, except for the highest conductance state (HCS) and the lowest conductance state (LCS). (c) Decoding fidelity of a distance-3 surface code for a typical physical fault rate of 10^{-2} using a trained recurrent neural network (RNN), as a function programming variability of the weights, represented as a factor of the polynomial fit used in (b). The error bands represent the 95% confidence interval over 10 random seeds. The performance of the RNN remains close to the digital baseline before the noise standard deviation reaches a factor of $\sim$10 times higher than the experimental value, after which the fidelity drops significantly.

characterization process is detailed in supplementary information note S3. In this model, the actual programmed conductance values are expressed as

$$G_{jk}^{\pm} \leftarrow G_{jk}^{\pm} + \mathcal{N}\left(0, \sigma_{\mathrm{prog}}\left(G_{jk}^{\pm}\right)\right), \tag{3}$$

where $\sigma_{\mathrm{prog}}(G_{jk}^{\pm})$ follows the polynomial fit of figure 4(b). The median value of the relative variations, that is, $\sigma_{\mathrm{prog}}(G_{jk}^{\pm})/G_{jk}^{\pm}$, is 0.8% for TiO$_x$-based resistive memory devices in the conductance range of [60, 200] μS.

The device yield of a chip is usually slightly under 100% due to variations and imperfections in the nanofabrication process. This translates to a non-zero probability of stuck-at fault devices. In this case, the device cannot be programmed to the desired conductance for mapping a given weight of the NN. Recent work on passive crossbars of memristors has shown a probability of stuck-at fault devices on the order of $\sim$1% [68–70]. In the case of our TiO$_x$-based resistive memory devices, this probability can reach $\sim$10% and the devices are usually stuck in their HCS.

4. Results

Due to the hardware non-idealities characterized in the previous section, it is expected that the transfer of a digitally trained neural decoder to the equivalent memristor-based implementation would decrease its performance. We simulate the impact of these key non-idealities on the fidelity of an MND implemented in a mixed-signal IMC architecture. We then benchmark mitigation strategies using re-training to almost completely compensate for the negative impact of the non-idealities of memristors.

We first evaluate the impact of the programming variability observed in TiO$_x$-based memristive devices by investigating their effects on the decoder performance. Figure 4(c) shows the evolution of the decoding fidelity of a distance-3 RNN for a typical physical fault rate of 10^{-2} when the programming noise is increased. The experiment is repeated 10 times to obtain a mean fidelity value (shown using a blue curve). Note that no noticeable fidelity decrease is observed below about 10 times the experimental noise value, therefore, the programming variability of TiO$_{\mathbf{x}}$-based resistive memory devices does not represent a roadblock for the realization of the MND. Evaluating the impact of the resolution of the DACs and ADCs (see supplementary information figure S1) leads to the same conclusion, as an 8-bit discretization of the input and output values does not present any significant impact on the performance of the decoding. Lastly, the neural decoder appears to be robust against the reading variability (up to 1%) induced by the analog electronics during a MVM operation (see figure S2). The methodology used to evaluate the impact of the DACs and ADCs, and the reading variability is described in supplementary information note S2.

We then study the impact of stuck-at fault devices on the fidelity of the neural decoder, which is expected to reach to up to 10% for the evaluated memristor technology. To emulate the impact of this non-ideality, a random subset of network parameters is set to zero when testing the decoder. This situation corresponds to

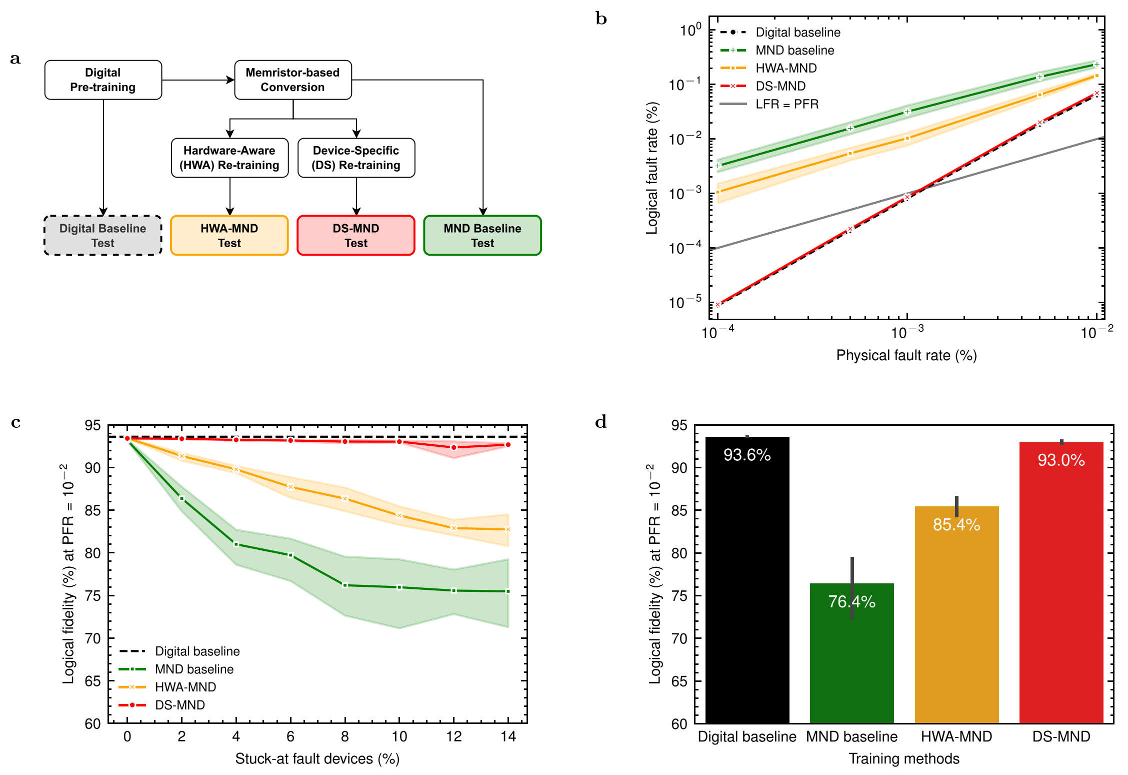

Figure 5. Baselines and re-training methods for the memristive neural decoder. (a) Schematic representation of the methods used to train the recurrent neural network (RNN) decoder. The fidelity of the digital baseline is tested immediately after classical training. The hardware-aware (HWA) and device-specific (DS) memristive neural decoder (MND) methods benefit from a re-training step after being converted to a memristor-based model with the *IBM Analog Hardware Acceleration Kit* [71], whereas the MND baseline training method is converted and tested without any re-training. (b) logical fault rate (LFR) after decoding as a function of the physical fault rate (PFR). The intersection between the line 'LFR = physical fault rate (PFR)' and decoder performances represent the breakeven points (i.e. the pseudo-threshold values) of the corresponding neural decoders. (c) Logical fidelity percentage of the decoder at a PFR of 10^{-2}, for the baselines and the re-trained decoders, as a function of the percentage of stuck-at fault devices. For the HWA-MND, the stuck-at fault devices define the dropconnect rate used during re-training. (d) Logical fidelity percentage of the decoder at a PFR of 10^{-2} for the baselines and the re-trained decoders. For the results presented in (b) and (d), the rate of stuck-at fault devices is set at the expected fabrication yield of 10%. For the results presented in (b), (c), and (d), the digital baseline fidelity is obtained with an equivalent neural decoder. The error bars and shaded regions represent the 95% confidence interval over 10 random seeds.

the case where one of the memristor of a differential pair is stuck in HCS or LCS and the other is either also stuck in the same state or purposefully programmed in order to bring the logical analog weight to zero. The results presented in figure 5(c) (green curve) show that, without any re-training designed to tackle this non-ideality, the stuck-at fault rate significantly reduces the performance of the decoder. A decrease of more than 17% in fidelity is observed when the percentage of stuck-at fault devices during inference reaches 8% (this is equivalent to 15% of the model parameters, since a parameter is encoded via a pair of devices).

Therefore, we explore two re-training methods to attempt to restore the initial fidelity of the MND (see figure 5(a)), to which we compare two baselines:

- Digital baseline: an ideal version of the neural decoder that can perform syndrome processing at room temperature on classical hardware. The parameters are trained using a deep learning method [72] (see supplementary information table S1) and are represented as 32-bit floating-point variables that do not include hardware non-idealities. The RNN architecture is based on [20]. The digital baseline performance is represented in black in figure 5, which we use as an upper bound and a reference against which to compare the other methods.
- MND baseline: a naïve implementation of an analog memristor-based neural decoder that could be integrated in a cryogenic environment. The RNN's architecture is identical to the digital baseline, but the parameters are converted into the equivalent memristor conductance values after digital training. Hardware constraints and memristor non-idealities are simulated based on experimental characterizations (see supplementary information table S2), but no re-training is used. The MND baseline performance is represented in green in figure 5, which is expected to be a lower bound for other MND methods.
- Hardware-aware memristive neural decoder (HWA-MND): the MND baseline augmented with a re-training post-processing. The re-training consists of one additional training epoch during which different random

weights are set to 0 at each forward pass. The dropconnect method [73] is typically used for regularization, but in the context of HWA re-training it is incorporated to improve the robustness of the RNN against stuck-at fault devices. For this reason, we set the dropconnect rate to match the expected number of weights blocked due to stuck-at fault devices in the characterized hardware (see supplementary information figure S4 for the different stuck-at fault rates). At the testing stage, the hardware non-idealities are simulated (see supplementary information table S2). The performance of HWA-MND is represented in yellow in figure 5.
- Device-specific memristive neural decoder method (DS-MND): a variation of the HWA-MND, where the dropconnect is not random but follows a specific device. During the entire re-training epoch, the weights corresponding to the stuck-at fault memristors in a given crossbar are fixed at 0. This approach forces the RNN to precisely adapt its parameters to this hardware limitation. The performance of DS-MND is represented in red in figure 5.

supplementary information note S4 provides details about the implementation of training and inference in our experiments, and we have made the simulation data available in [74].

Dropping connections randomly during the HWA-MND re-training provides a generic mitigation strategy applicable to any crossbar of memristors given that the average number of stuck-at fault devices is known prior to re-training. However, it does not allow for a full recovery of the digital baseline performance even for low rates of stuck-at fault devices. DS-MND addresses this shortcoming. This re-training of DS-MND completely avoids updating weights that cannot be reliably programmed in the crossbar. The idea of this DS re-training is to leverage precise knowledge of the crossbars' electrical characterizations to find the best NN parameters for that particular memristive circuit. It has the advantage of converging towards a nearly optimal solution, at the cost of having to individually characterize each crossbar of the MND to localize stuck-at fault devices, a task which can be performed through methods such as march tests [75, 76]. The simulation results, with the expected 10% rate of stuck-at fault devices (see figure 5(d)), show that only DS-MND reaches within 1% of the digital baseline's fidelity, whereas the HWA-MND recovers only about half of the performance loss due to hardware non-idealities.

Figure 5(b) represents the decoding performance of the MNDs studied in the case of different PFRs for a fixed percentage of stuck-at fault devices of 10% to match the typical fabrication yield. Only the DS re-training method maintains the pseudo-threshold of the memristive decoder near the pseudo-threshold of the baseline decoder. Therefore, based on the characterized non-idealities of TiO_x-based resistive memory devices, it is a necessity to introduce specific knowledge of the chip during re-training to achieve the highest decoding performance in the case of a distance-3 code based on the RNN we have studied. The general solution provided by HWA-MND provides robustness against device-specific hardware issues, but remains insufficient.

5. Discussion

We implemented a simulation based on the *IBM Analog Hardware Acceleration Kit* [71] that accounts for the experimentally characterized non-idealities of TiO_x-based resistive memory devices and measured their impact on our neural decoder's performance. By applying computational methods to mitigate key hardware non-idealities, we improved the robustness of the neural decoder. In particular, we found that using the dropconnect method during re-training can greatly improve the fidelity of an MND that whose performance is significantly reduced by stuck-at fault devices. Moreover, we have seen that localizing stuck-at fault memory devices in the crossbars and using that knowledge to disable the corresponding connections during re-training can lead to a fidelity score comparable to that of the digital baseline (i.e. having a $<1\%$ difference). Furthermore, the MND exhibits only a small drop in the pseudo-threshold for the distance-3 surface code in numerical simulations (see figure 5(b)). Therefore, we can conclude that our results support the effectiveness of specialized training methods for MNDs to achieve near-optimal performance.

Although an experimental proof of concept has not yet been performed on fully integrated hardware, the results we have presented offer a promising pathway to realizing high-fidelity neural decoders using IMC and analog memristive devices. One interesting avenue for future research is the development of a fully analog version of the memristive decoder circuit presented in this paper. Such an implementation would bring further benefits in terms of the decoding time and energy efficiency. Analog activation functions have been reported in the literature [77]. For instance, the ReLU function can be applied in the analog domain using multimodal transistors [78]. It is therefore possible to envision MNDs using an analog activation function after the first layer instead of implementing an ADC followed by a digital activation function followed by a DAC, as presented in this paper. A practical circuit would also necessitate many more electronic components to realistically perform the decoding task, some of which we did not consider here. For example, transimpedance amplifiers (TIAs) would be needed to convert a current signal to a voltage signal at the output of each memristor row. Also, an analog memory unit might be necessary to store and transmit the hidden states' signals back to the recurrence input ports.

In our work, we made the assumption that imperfections arising from analog or digital CMOS components (e.g. noise introduced by differential amplifiers, TIAs, and ADCs/DACs; divergence of the analog activation function from its digital counterpart in the case of a fully analog implementation; and signal distortion arising from multiple recurrences) are much less critical in terms of the fidelity of the MND in comparison to the non-idealities exhibited by the memristors. However, future work should include a more exhaustive analysis of the impact of circuit-level imperfections (e.g. using training methodologies that have been introduced [79, 80]).

Scalability is a well-known issue for QEC decoders: as the error correction code distance increases, the syndrome space grows exponentially. Neural decoders thus require an exponentially larger dataset to learn about correlations between syndromes and the occurrences of logical errors. Our decoder cannot overcome the issue of scalability. However, it can be integrated in a multi-stage or hierarchical decoding scheme [54, 55, 81]. Such approaches have been proposed to improve the efficiency of NN-based decoders by using a combination of two decoding modules; in some instances [82], the first module is a simple classical decoder, and the second is a NN-based decoder. The role of the NN-based module is to act as a supervisor, identifying when the correction suggested by the simple decoder will lead to a logical error. This approach results in a constant execution time once the NN has been trained, regardless of the physical error rate, and scales linearly with the number of qubits in the code. A more recent study [83] demonstrates that NNs can be used in the first stage, as local decoders, to remove an initial set of errors, thereby reducing the syndrome space and enabling the fast execution of a global decoder (minimum-weight perfect matching in the study) to correct the remaining errors. In this sense, a fully analog version of our MND could be integrated in a hierarchical decoding strategy and act as a local decoder to feed inputs to a global decoder.

Another challenge related to the scalability of decoders is the cryogenic compatibility of the chosen hardware. Indeed, as the number of physical qubits increases, to avoid a wiring bottleneck between the control electronics at room temperature and the quantum processor in a cryogenic environment, it is highly beneficial to integrate the decoder hardware directly within the cryostat [6]. Within this scope, the energy consumption and thus the heat dissipation of the decoder should be minimized to avoid perturbations in the quantum system. The MND presented in this paper shows promise in terms of cryogenic compatibility as the MAC operations rely on an energy-efficient memristive IMC architecture instead of digital circuit blocks such as multipliers and adders. Even if current hardware implementations of NNs employing application-specific integrated circuits and field-programmable gate arrays are not optimized in comparison to CPU-based approaches, they continue to suffer from the delays and energy expenditure associated with digital MAC operations. Furthermore, in recently proposed approaches introducing the idea of quantized IMC for QEC [36, 37], the implementations still require digital multipliers and adders. From their fast inference time and energy efficiency, MNDs are a promising technology for direct integration in a dilution refrigerator. However, the cryogenic compatibility of a memristor-based fully analog integrated circuit for QEC and the detailed characterization of its decoding time and power dissipation remains to be investigated.

Data availability statement

The code from our study is available from the corresponding author upon reasonable request. The syndromes dataset has been made publicly available [56] as well as the model training and simulation output data [74].

The data that support the findings of this study are openly available at the following URL/DOI: https://doi.org/10.5281/zenodo.11166209.

Acknowledgments

We thank our editor, Marko Bucyk, for his careful review and editing of the manuscript. The authors acknowledge the financial support received through the NSF's CIM Expeditions award (CCF-1918549). P R acknowledges the financial support of Mike and Ophelia Lazaridis, Innovation, Science and Economic Development Canada (ISED), and the Perimeter Institute for Theoretical Physics. Research at the Perimeter Institute is supported in part by the Government of Canada through ISED and by the Province of Ontario through the Ministry of Colleges and Universities. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). LN2 is a French-Canadian joint International Research Laboratory (IRL-3463) funded and co-operated by the Centre National de la Recherche Scientifique (CNRS), the Université de Sherbrooke, the Université de Grenoble Alpes (UGA), the école Centrale de Lyon (ECL), and the Institut National des Sciences Appliquées de Lyon (INSA Lyon). It is supported by the Fonds de Recherche du Québec—Nature et Technologie (FRQNT).

Author contributions

All authors contributed to this article and approved of the submitted version.

Victor Yon: methodology, software implementation, run simulations, results analysis and visualization, writing–original draft preparation.

Frédéric Marcotte: methodology, software implementation, run simulations, hardware characterization, writing–original draft preparation.

Pierre-Antoine Mouny: methodology, software implementation, hardware characterization, manuscript editing.

Gebremedhin A. Dagnew: methodology, syndrome dataset generation, manuscript editing.

Bohdan Kulchytskyy: syndrome dataset generation, manuscript review, supervision.

Sophie Rochette: methodology, manuscript editing, supervision.

Yann Beilliard: methodology, manuscript editing, supervision.

Dominique Drouin: manuscript review, supervision.

Pooya Ronagh: methodology, manuscript editing, supervision.

Conflict of interests

The authors declare no competing interests.

ORCID iDs

Victor Yon https://orcid.org/0000-0003-4517-5042
Pierre-Antoine Mouny https://orcid.org/0000-0003-4941-8347
Yann Beilliard https://orcid.org/0000-0003-0311-8840
Dominique Drouin https://orcid.org/0000-0003-2156-967X
Pooya Ronagh https://orcid.org/0000-0002-9591-9727

References

[1] Shor P W 1994 *Proc. 35th Annual Symposium on Foundations of Computer Science* (Ieee) pp 124–34
[2] Gidney C and Ekerå M 2021 *Quantum* **5** 433
[3] Preskill J 1998 *Proc. R. Soc.* A **454** 385
[4] Beverland M E, Murali P, Troyer M, Svore K M, Hoefler T, Kliuchnikov V, Low G H, Soeken M, Sundaram A and Vaschillo A 2022 (arXiv:2211.07629)
[5] Camps D, Rrapaj E, Klymko K, Austin B and Wright N J 2024 *ISC High Performance 2024 Research paper Proc. (39th Int. Conf.)* (Prometeus GmbH) pp 1–12
[6] Reilly D J 2019 *2019 IEEE Int. Electron Devices Meeting (IEDM)* p 31.7.1–31.7.6
[7] Roffe J 2019 *Contemp. Phys.* **60** 226
[8] Battistel F, Chamberland C, Johar K, Overwater R W J, Sebastiano F, Skoric L, Ueno Y and Usman M 2023 Real-time decoding for fault-tolerant quantum computing: progress, challenges and outlook (arXiv:2303.00054)
[9] Holmes A, Jokar M R, Pasandi G, Ding Y, Pedram M and Chong F T 2020 NISQ+: boosting quantum computing power by approximating quantum error correction (arXiv:2004.04794)
[10] Ueno Y, Kondo M, Tanaka M, Suzuki Y and Tabuchi Y 2021 *2021 58th ACM/IEEE Design Automation Conf. (DAC)* (Ieee) pp 451–6
[11] Ueno Y, Kondo M, Tanaka M, Suzuki Y and Tabuchi Y 2022 *2022 IEEE Int. Symp. on High-Performance Computer Architecture (HPCA)* (IEEE) pp 274–87
[12] Delfosse N and Nickerson N H 2021 *Quantum* **5** 595
[13] Wu Y, Liyanage N and Zhong L 2022 An interpretation of union-find decoder on weighted graphs (arXiv:2211.03288)
[14] Chamberland C, Kubica A, Yoder T J and Zhu G 2020 *New J. Phys.* **22** 023019
[15] Das P, Pattison C A, Manne S, Carmean D M, Svore K M, Qureshi M and Delfosse N 2022 *2022 IEEE Int. Symp. on High-Performance Computer Architecture (HPCA)* pp 259–73
[16] Liyanage N, Wu Y, Tagare S and Zhong L 2024 arXiv:2406.08491
[17] Tan X, Zhang F, Chao R, Shi Y and Chen J 2023 *PRX Quantum* **4** 040344
[18] Skoric L, Browne D E, Barnes K M, Gillespie N I and Campbell E T 2023 *Nat. Commun.* **14** 7040
[19] Bombín H, Dawson C, Liu Y-H, Nickerson N, Pastawski F and Roberts S 2023 arXiv:2303.04846
[20] Chamberland C and Ronagh P 2018 *Quantum Sci. Technol.* **3** 044002
[21] Verma N, Jia H, Valavi H, Tang Y, Ozatay M, Chen L-Y, Zhang B and Deaville P 2019 *IEEE Solid-State Circuits Mag.* **11** 43
[22] El Mesoudy A, Lamri G, Dawant R, Arias-Zapata J, Gliech P, Beilliard Y, Ecoffey S, Ruediger A, Alibart F and Drouin D 2022 *Microelectron. Eng.* **255** 111706
[23] Berggren K *et al* 2020 *Nanotechnology* **32** 012002
[24] Chua L 1971 *IEEE Trans. Circuit Theory* **18** 507
[25] Strukov D B, Snider G S, Stewart D R and Williams R S 2008 *Nature* **453** 80
[26] Chua L 2011 *Appl. Phys.* A **102** 765
[27] Song M-K *et al* 2023 *ACS Nano* **17** 11994
[28] Kumar D, Aluguri R, Chand U and Tseng T 2017 *Ceram. Int.* **43** S547
[29] Woo J and Yu S 2018 *IEEE Nanotechnol. Mag.* **12** 36

[30] Sebastian A, Le Gallo M, Khaddam-Aljameh R and Eleftheriou E 2020 *Nat. Nanotechnol.* **15** 529
[31] Amirsoleimani A, Alibart F, Yon V, Xu J, Pazhouhandeh M R, Ecoffey S, Beilliard Y, Genov R and Drouin D 2020 *Adv. Intell. Syst.* **2** 2000115
[32] Hu M, Strachan J P, Li Z, Grafals E M, Davila N, Graves C, Lam S, Ge N, Yang J J and Williams R S 2016 *2016 53nd ACM/EDAC/IEEE Design Automation Conf. (DAC)* pp 1–6
[33] Beilliard Y, Paquette F, Brousseau F, Ecoffey S, Alibart F and Drouin D 2020 *Nanotechnology* **31** 445205
[34] Mouny P-A *et al* 2023 *IEEE Trans. Electron Devices* **70** 1989
[35] Mouny P-A, Dawant R, Galaup B, Ecoffey S, Pioro-Ladrière M, Beilliard Y and Drouin D 2023 *Appl. Phys. Lett.* **123** 163505
[36] Wang P, Peng X, Chakraborty W, Khan A I, Datta S and Yu S 2020 *2020 IEEE Int. Electron Devices Meeting (IEDM)* pp 38.5.1–4
[37] Ichikawa Y, Goda A, Matsui C and Takeuchi K 2022 *2022 IEEE Int. Memory Workshop (IMW)* pp 1–4
[38] Ali H, Marques J, Crawford O, Majaniemi J, Serra-Peralta M, Byfield D, Varbanov B, Terhal B M, DiCarlo L and Campbell E T 2024 arXiv:2403.00706
[39] Liyanage N, Wu Y, Deters A and Zhong L 2023 Scalable quantum error correction for surface codes using FPGA (arXiv:2301.08419)
[40] Barber B et al 2023 A real-time, scalable, fast and highly resource efficient decoder for a quantum computer (arXiv:2309.05558)
[41] Mouny P-A, Benhouria M, Yon V, Dufour P, Huang L, Beilliard Y, Rochette S, Drouin D and Ronagh P 2024 *2024 IEEE Int. Conf. on Quantum Computing and Engineering (QCE)* (IEEE) pp 1258–63
[42] Caune L et al 2024 Demonstrating real-time and low-latency quantum error correction with superconducting qubits (arXiv:2410.05202)
[43] Stillmaker A and Baas B 2017 *Integration* **58** 74–81
[44] Joshi V, Le Gallo M, Haefeli S, Boybat I, Nandakumar S R, Piveteau C, Dazzi M, Rajendran B, Sebastian A and Eleftheriou E 2020 *Nat. Commun.* **11** 2473
[45] Wang C *et al* 2019 *ACM Trans. Des. Autom. Electron. Syst.* **24** 1–37
[46] Xi Y, Gao B, Tang J, Chen A, Chang M-F, Hu X S, Spiegel J V D, Qian H and Wu H 2021 *Proc. IEEE* **109** 14–42
[47] Chakraborty I, Ali M, Ankit A, Jain S, Roy S, Sridharan S, Agrawal A, Raghunathan A and Roy K 2020 *Proc. IEEE* **108** 2276–310
[48] Yon V, Amirsoleimani A, Alibart F, Melko R G, Drouin D and Beilliard Y 2022 *Front. Electron.* **3** 825077
[49] Bombin H and Martin-Delgado M A 2007 *Phys. Rev.* A **76** 012305
[50] Horsman C, Fowler A G, Devitt S and Van Meter R 2012 *New J. Phys.* **14** 123011
[51] Tomita Y and Svore K M 2014 *Phys. Rev.* A **90** 062320
[52] Baireuther P, O'Brien T E, Tarasinski B and Beenakker C W J 2018 *Quantum* **2** 48
[53] Litinski D 2019 *Quantum* **3** 128
[54] Varsamopoulos S, Bertels K and Almudever C G 2020 *Quantum Mach. Intell.* **2** 3
[55] Delfosse N 2020 Hierarchical decoding to reduce hardware requirements for quantum computing (arXiv:2001.11427)
[56] Yon V and Dagnew G 2024 A memristive neural decoder for cryogenic fault-tolerant quantum error correction - syndromes dataset *Zenodo* (https://doi.org/10.5281/zenodo.11166209)
[57] Chamberland C and Beverland M E 2018 *Quantum* **2** 53
[58] Conrad J, Chamberland C, Breuckmann N P and Terhal B 2018 *Phil. Trans. R. Soc.* A **376** 20170323
[59] Fowler A G, Whiteside A C and Hollenberg L C L 2012 *Phys. Rev.* A **86** 042313
[60] Mohseni M et al 2024 How to build a quantum supercomputer: scaling challenges and opportunities (arXiv:2411.10406)
[61] Gokmen T, Rasch M J and Haensch W 2018 *Front. Neurosci.* **12** 745
[62] Milo V, Malavena G, Monzio Compagnoni C and Ielmini D 2020 *Materials* **13** 166
[63] Alibart F, Gao L, Hoskins B D and Strukov D B 2012 *Nanotechnology* **23** 075201
[64] Ielmini D and Wong H-S P 2018 *Nat. Electron.* **1** 333
[65] Zhang Y, Huang P, Gao B, Kang J and Wu H 2020 *J. Appl. Phys.* **54** 083002
[66] Adam G C, Khiat A and Prodromakis T 2018 *Nat. Commun.* **9** 5267
[67] Chen C-Y, Shih H-C, Wu C-W, Lin C-H, Chiu P-F, Sheu S-S and Chen F T 2015 *IEEE Trans. Comput.* **64** 180
[68] Kim H, Mahmoodi M R, Nili H and Strukov D B 2021 *Nat. Commun.* **12** 5198
[69] Jang J, Gi S, Yeo I, Choi S, Jang S, Ham S, Lee B and Wang G 2022 *Adv. Sci.* **9** 2201117
[70] Yeon H *et al* 2020 *Nat. Nanotechnol.* **15** 574
[71] Rasch M J, Moreda D, Gokmen T, Le Gallo M, Carta F, Goldberg C, El Maghraoui K, Sebastian A and Narayanan V 2021 *2021 IEEE 3rd Int. Conf. on Artificial Intelligence Circuits and Systems (AICAS)* pp 1–4
[72] LeCun Y, Bengio Y and Hinton G 2015 *Nature* **521** 436
[73] Wan L, Zeiler M, Zhang S, Le Cun Y and Fergus R 2013 *Proc. 30th Int. Conf. on Machine Learning, (Atlanta, Georgia, USA), (Proc. Machine Learning Research)* vol 28, ed S Dasgupta and D McAllester (Pmlr) pp 1058–66
[74] Yon V 2024 A memristive neural decoder for cryogenic fault-tolerant quantum error correction - simulation data *Zenodo* (https://doi.org/10.5281/zenodo.11164068)
[75] van de Goor A and Zorian Y 1993 *1993 European Conf. on Design Automation With the European Event in ASIC Design* pp 499–505
[76] Chen Y-X and Li J-F 2015 *2015 IEEE 33rd VLSI Test Symp. (VTS)* pp 1–6
[77] Krestinskaya O, Choubey B and James A P 2020 *Sci. Rep.* **10** 5838
[78] Surekcigil Pesch I, Bestelink E, de Sagazan O, Mehonic A and Sporea R A 2022 *Sci. Rep.* **12** 670
[79] Liu B, Li H, Chen Y, Li X, Huang T, Wu Q and Barnell M 2014 *2014 IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)* pp 63–70
[80] Liu B, Li H, Chen Y, Li X, Wu Q and Huang T 2015 *2015 52nd ACM/EDAC/IEEE Design Automation Conf. (DAC)* pp 1–6
[81] Meinerz K, Park C-Y and Trebst S 2022 *Phys. Rev. Lett.* **128** 080505
[82] Varsamopoulos S, Bertels K and Almudever C G 2020 *IEEE Trans. Comput.* **69** 300
[83] Chamberland C, Goncalves L, Sivarajah P, Peterson E and Grimberg S 2022 Techniques for combining fast local decoders with global decoders under circuit-level noise (arXiv:2208.01178)