Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Variability-Aware Memristor Circuit for Hierarchical Bayesian Temporal Sampling

Tarcisius Januel
Aix Marseille Univ, Université de Toulon
Tifenn Hirtzlin
CEA, LETI, Université Grenoble-Alpes
Olivier Billoint
CEA, LETI, Université Grenoble-Alpes
Djohan Bonnet
CEA, LETI, Université Grenoble-Alpes
Louis Hutin
CEA, LETI, Université Grenoble-Alpes
Damien Querlioz
Université Paris-Saclay, CNRS, Centre de Nanosciences et de Nanotechnologies
Jean-Michel Portal
Aix Marseille Univ, Université de Toulon
Elisa Vianello
elisa.vianello@cea.fr

CEA, LETI, Université Grenoble-Alpes

Article

Keywords:

Posted Date: September 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11061798/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: Competing interest reported. Damien Querlioz is a Guest Editor of the Collection Future Computing: Paradigms and Design for Next Generation Applications. Damien Querlioz was not involved in the journal's review of, or decisions related to, this paper. The remaining authors declare no competing interests.

Variability-Aware Memristor Circuit for Hierarchical Bayesian Temporal Sampling

Tarcisius Januel[1,2], Tifenn Hirtzlin[2], Olivier Billoint[2], Djohan Bonnet[2], Louis Hutin[2], Damien Querlioz[3], Jean-Michel Portal[1], and Elisa Vianello[2,*]

[1]Aix Marseille Univ, Université de Toulon, CNRS, IM2NP, Marseille, France
[2]CEA, LETI, Université Grenoble-Alpes, 38400 Grenoble, France
[3]Université Paris-Saclay, CNRS, Centre de Nanosciences et de Nanotechnologies, 91120 Palaiseau, France
[*]elisa.vianello@cea.fr

ABSTRACT

Bayesian neural networks (BNNs) require hardware capable of storing and sampling probabilistic synaptic weights. While population coding exploits device-to-device (D2D) variability to encode weight distributions, temporal sampling instead requires controlled and reproducible stochastic fluctuations, making its hardware implementation particularly challenging. Here, we demonstrate a memristor-based temporal-coding circuit that combines three analog memristor devices per synaptic weight to encode its mean and standard deviation with an externally injected white-noise source for sample generation. A dedicated fluctuation-amplification stage provides controllable temporal stochasticity, while a hierarchical training algorithm explicitly models device-to-device variability and temporal noise to ensure robust Bayesian inference under hardware non-idealities. On the MIT-BIH arrhythmia classification task, the proposed approach achieves a 2.7-fold reduction in memory footprint compared with an equivalent population-coding implementation, while preserving Bayesian inference accuracy and uncertainty estimates.

Introduction

Bayesian neural networks (BNNs) provide a principled framework for quantifying uncertainty in artificial intelligence by representing each synaptic weight as a probability distribution rather than a deterministic value[1–3]. Accurate inference therefore requires repeated sampling from these distributions, placing stringent demands on hardware architectures for the efficient storage and generation of stochastic weights[4]. Beyond conventional FPGA-based implementations[5], compute-in-memory (CIM) architectures offer a promising alternative. In particular, the probabilistic nature of Bayesian inference can be mapped naturally onto analog CIM with non-volatile memories (NVMs), turning the intrinsic variability of physical devices from a source of uncertainty into a computational resource[6,7].

A classical approach is population coding, in which a probability distribution is encoded by a population of nominally identical memory devices that exploit their intrinsic device-to-device (D2D) variability[8]. Although this approach provides a direct representation of uncertainty, each synaptic weight requires as many devices as samples, resulting in substantial memory overhead that limits scalability.

Temporal sampling provides an alternative to population coding by using a single bit-cell as a stochastic variable that is repeatedly programmed or read to generate samples from its intrinsic cycle-to-cycle (C2C)[9,10] or read-to-read (R2R) variability[11–16]. Temporal stochasticity can arise either during programming or readout. Write-based approaches exploit C2C fluctuations in devices whose conductance is highly sensitive to charge trapping, such as two-dimensional material-based memories[9]. However, repeated programming progressively consumes device endurance, limiting the number of samples that can be generated over the device lifetime. Read-based approaches instead harvest intrinsic read noise, for example from the RESET state of memristors[17]. However, mitigating D2D variability in such architectures requires the parallel readout of multiple cells, partially offsetting the footprint advantage of temporal coding[11]. Temporal sampling can thus substantially reduce the

memory overhead of Bayesian hardware, but realizing its full potential requires stochastic fluctuations that are sufficiently strong, controllable and reproducible across devices while remaining compatible with compact single-cell implementations. A comparative schematic summarizing the main characteristics of population and temporal sampling is provided in Supplementary Note 1.

Here, we introduce a memristor-based temporal-coding circuit that combines analog storage of the Gaussian mean μ and standard deviation σ with externally injected white noise. This architecture addresses two key requirements for scalable hardware sampling. First, independent programming of the mean and standard deviation enables orthogonal exploration of the Gaussian parameter space. This capability has also been pursued in recent approaches based on a ferroelectric NAND cell, in which the Gaussian distribution is shaped during programming through voltage-step-controlled write operations[13], and on a dual-channel memristor with intrinsically decoupled conductance channels[14]. In contrast, our architecture separates the non-volatile storage of (μ, σ) from the generation of stochasticity during readout, enabling independent control of the two parameters using a standard CMOS-compatible hafnium-oxide-based memristor technology integrated in the BEOL of a mature process, while avoiding repeated write cycles for distribution shaping. Second, rather than relying on a single entropy source per layer, as in previous approaches based on 3D Fe-diode devices[15,16], our architecture distributes an externally generated white-noise source across the sampling circuit, enabling the generation of uncorrelated samples at frequencies above 1 kHz.

In this work, we develop a fully custom temporal-sampling demonstrator to experimentally assess the use of externally injected white noise for memristor-based Bayesian inference. We first show that intrinsic memristor fluctuations do not provide the controllable and spectrally white stochasticity required for temporal sampling, motivating the separation of non-volatile parameter storage from temporal noise generation. We then evaluate the resulting hardware-aware framework at the network level through hardware-calibrated simulations. By decoupling non-volatile storage from temporal stochasticity and explicitly incorporating device variability into learning, this work establishes a route to compact Bayesian inference with memristive devices, in which the number of stochastic samples can be increased in time without increasing the non-volatile memory footprint. Details of chip fabrication, electrical characterization and hierarchical training are provided in the Methods.

Results

Memristor-based circuit for Bayesian temporal sampling

Read-based temporal coding poses a stringent challenge for NVM-based hardware implementations becauses representing each synaptic weight with a single programmed conductance requires precise control of the programmed state. In a Bayesian representation, each synaptic weight is described by independent mean and standard-deviation parameters, μ and σ, whose accurate encoding directly determines the statistics of the generated temporal samples. We therefore decouple the deterministic and stochastic components of the weight representation by introducing an external temporal Gaussian random-number generator (GRNG). The GRNG provides a controllable and reproducible external variance, σ^2_{ext}, thereby relaxing the precision required from the NVM devices to directly encode of the stochastic component.

To experimentally demonstrate this approach with memristor technology, we fabricated a 2048-memristors demonstrator implementing external white-noise injection (Fig. 1a). The array comprises 64×32 memristors, addressed through word-line (WL), source-line (SL) and bit-line (BL) shift registers, with BL capacitors used for the temporal accumulation of the cell outputs. The memristor stack consists of TiN/Ti/HfO_x/TiN and is integrated in the back-end-of-line (BEOL) of a 130 nm CMOS technology. Its integration is evidenced by the FIB cross-section in Fig. 1b, which shows the memristor devices above the M1–M4 metal interconnect levels and the underlying transistor layer, and by the optical micrograph in Fig. 1c, which shows the 64×32 array together with the WL, SL and BL shift registers, BL capacitors and charge-to-voltage conversion circuitry based on a charge trans-impedance amplifier (CTIA).

Each synaptic weight is encoded using three memristor devices, separating the control of its mean and standard deviation. One device defines the stochastic component through the conductance G_σ, while two differential devices encode the positive and negative contributions, μ_+ and μ_-. This three-device representation enables μ and σ to be programmed independently,

while retaining a compact cell-level implementation. The operating principle is illustrated in the fabricated single-bit cell of Fig. 1d. The σ branch comprises a memristor device with conductance G_σ, an RC filtering node and a source follower transistor T1. External white noise is injected through selector transistor T2; G_σ controls the effective RC cutoff frequency and therefore the amplitude of the fluctuations reaching the output. The two additional one transistor - one memristor branches independently generate the positive and negative mean currents, $i_{\mu+}$ and $i_{\mu-}$. These three contributions are summed at the CTIA, producing successive temporal output samples that follow the encoded Gaussian distribution. After acquisition, the deterministic mean contribution is obtained from the difference between the averaged positive and negative branches, whereas the stochastic component is obtained from the fluctuations of the σ branch. The resulting output can therefore be expressed as

$$V_{out} = \langle V_{out,\mu+} \rangle - \langle V_{out,\mu-} \rangle + V_{out,\sigma} - \langle V_{out,\sigma} \rangle , \tag{1}$$

where the first two terms define the encoded mean, μ, and the last two terms form a zero-mean stochastic contribution with variance σ^2, such that $V_{out} \sim \mathcal{N}(\mu, \sigma^2)$. The complete readout therefore converts the spatially stored memristor conductances into a sequence of temporal samples, which are subsequently processed digitally.

The separate extraction procedures for σ and μ are detailed in Figs. 1e and 1f, respectively. For σ encoding (Fig. 1e), the memristor conductance G_σ is first programmed, after which white noise is injected into the cell and the resulting fluctuations are accumulated by the CTIA. The standard deviation of the output, std(V_{out}), is then extracted after removal of the DC component using 50 temporal samples. This procedure maps the programmed G_σ onto the effective output standard deviation and therefore onto σ. In contrast, μ encoding (Fig. 1f) relies on the differential branches: $G_{\mu+}$ and $G_{\mu-}$ are programmed independently, their contributions are accumulated separately by the CTIA, and the mean values extracted from 50 temporal samples are subtracted to obtain $\mu = \langle V_{out,\mu+} \rangle - \langle V_{out,\mu-} \rangle$. This separation of the programming and extraction paths allows the mean and variance of the temporal Gaussian distribution to be independently controlled at the cell level. The rationale for relying on an external entropy source rather than intrinsic memristor fluctuations is examined in the following section.

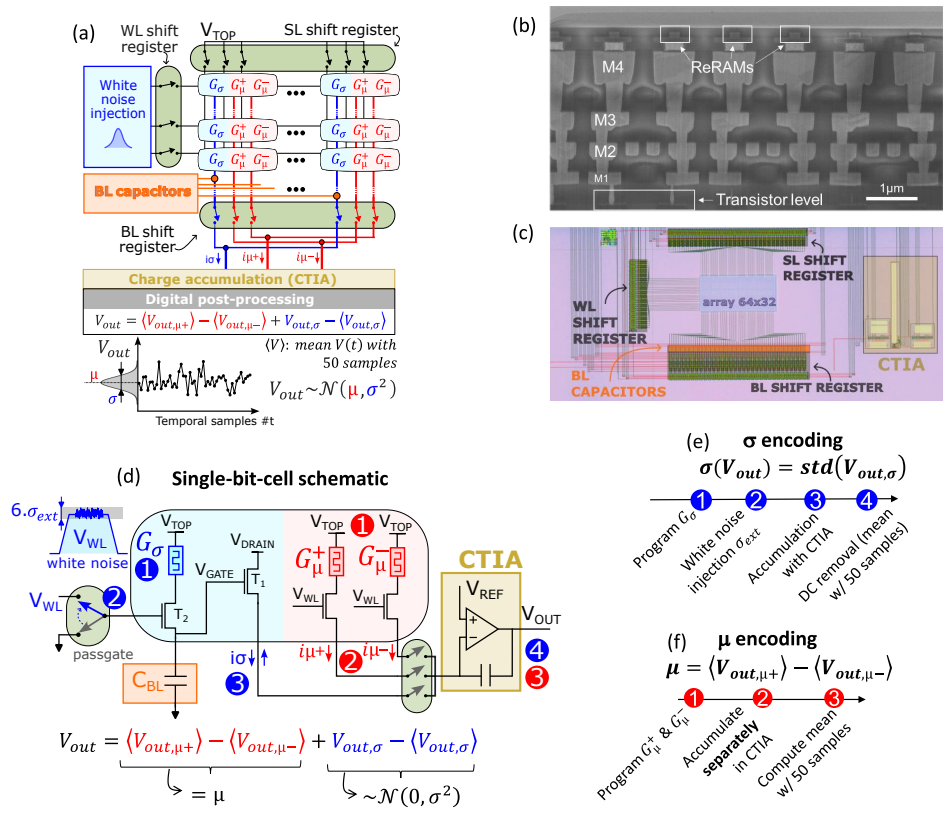


Figure 1. Design and operation of the memristor-based temporal Gaussian sampling demonstrator. a System-level schematic of the 2048 memristor demonstrator with external white-noise injection. The 64×32 array is addressed through WL, SL and BL shift registers. Each synaptic weight is encoded by three memristor devices: G_σ controls the stochastic component, whereas $G_{\mu+}$ and $G_{\mu-}$ differentially encode the mean. **b** FIB cross-section of the TiN/Ti/HfOx:Si/TiN memristor stack integrated in the BEOL of a 130 nm CMOS technology, showing the memristor devices above the M1–M4 metal levels and the underlying transistor layer. **c** Optical micrograph of the fabricated demonstrator, showing the 64×32 memristor array, WL, SL and BL shift registers, BL capacitors and the CTIA. **d** Schematic of the fabricated single-bit cell. The σ branch consists of the memristor device G_σ, an RC filtering node and source follower T1. External white noise is injected through selector transistor T2, with G_σ setting the effective RC cutoff frequency and hence the amplitude of the generated fluctuations. The two differential one transistor – one memristor branches independently encode μ_+ and μ_-. Their currents are combined with the stochastic contribution at the CTIA to generate successive samples of the encoded Gaussian distribution. **e** Experimental workflow for σ extraction: programming of G_σ, external white-noise injection, CTIA accumulation, and extraction of std(V_{out}) after DC removal using 50 temporal samples. **f** Experimental workflow for μ extraction: independent programming of $G_{\mu+}$ and $G_{\mu-}$, separate CTIA accumulation, and computation of the mean from 50 temporal samples according to $\mu = \langle V_{out,\mu+} \rangle - \langle V_{out,\mu-} \rangle$.

Intrinsic memristor variability limits temporal stochasticity

We next investigated whether the intrinsic variability of filamentary HfO_x/Ti memristors could provide the temporal stochasticity required for Bayesian sampling. We separately characterized device-to-device (D2D), cycle-to-cycle (C2C), and read-to-read (R2R) variability using dedicated programming and readout protocols (Fig. 2a). D2D variability was obtained by programming different devices once to the same target conductance and reading their programmed states, whereas C2C variability was evaluated by repeatedly programming the same device to the same target and reading each programmed state. R2R variability was instead measured by repeatedly reading a single programmed state without reprogramming. These measurements isolate, respectively, spatial variability between devices, programming-induced variability and temporal fluctuations during repeated reads.

For comparable target conductances, the three mechanisms produce markedly different distributions of conductance values (Fig. 2b). The extracted (μ_G, σ_G) pairs were obtained from 256 devices for D2D, 19 programming cycles on a single

device for C2C, and 50,000 reads of a single programmed state for R2R. The corresponding empirical distributions were approximated by normal distributions, $G \sim \mathcal{N}(\mu_G, \sigma_G^2)$, to quantify their respective spread. D2D measurements exhibit a relatively compact cluster in the (μ_G, σ_G) plane, whereas C2C measurements show a substantially broader distribution of both the mean conductance and its standard deviation. R2R measurements are characterized by much smaller conductance fluctuations for an individual programmed state, but these fluctuations vary strongly from device to device. Thus, even when devices are programmed to similar conductance values, their intrinsic temporal noise amplitudes are not uniquely determined by the programmed conductance.

This lack of control becomes apparent when the extracted standard deviation is plotted as a function of the mean conductance (Fig. 2c). D2D variability is substantially larger than R2R variability, with the latter being approximately five times smaller over the investigated conductance range. However, the R2R contribution exhibits a large device-to-device dispersion, preventing the noise amplitude from being reliably predicted from the conductance state. C2C variability also contributes a broad distribution, indicating that the stochastic response can change from one programming event to another. In addition, the mean R2R standard deviation remains approximately independent of the programmed conductance, in contrast to the required scaling of the stochastic component with the encoded σ. The observed R2R behaviour is consistent with previously reported conductance-fluctuation trends in filamentary oxide memristors[18].

Beyond its amplitude, the spectral character of the intrinsic fluctuations is also incompatible with the assumptions underlying temporal Gaussian sampling. The normalized R2R noise power spectral density (PSD), measured from 10,000 consecutive reads at $G = 130\ \mu$S, follows an approximate $1/f^{\beta}$ dependence (Fig. 2d), with $\beta \approx 1$ over the measured frequency range. This flicker-noise-like spectrum indicates temporally correlated fluctuations, in contrast to the frequency-independent PSD expected for white Gaussian noise ($\beta = 0$). Consequently, intrinsic memristor fluctuations provide neither a reproducible mapping between conductance and noise amplitude nor the spectrally white temporal randomness required by the sampling model.

These results motivate separating non-volatile parameter storage from temporal noise generation. In the proposed architecture, the memristor conductances encode the distribution parameters, while a dedicated external white-noise source provides the temporal randomness. The effect of this separation is illustrated in Fig. 2e, where the standard deviation of 512 temporal samples from the σ branch is plotted against its programmed conductance G_σ, with and without noise injection. Without injection, the measured $\sigma(V_{out})$ is determined by intrinsic R2R fluctuations and residual fluctuations from the measurement setup. The resulting standard deviation shows a broad and non-monotonic dependence on G_σ, preventing a reliable conductance-to-noise mapping. With external white-noise injection, in contrast, $\sigma(V_{out})$ increases approximately linearly with G_σ. The conductance therefore acts as a programmable gain for the externally generated stochastic signal, allowing the desired standard deviation to be set independently of the intrinsic variability of the memristor.

The spectral properties of the resulting output further demonstrate this separation (Fig. 2f). The normalized PSD of the σ branch, evaluated over 512 temporal samples, is approximately flat over the measured frequency range, corresponding to $\beta = 0$ and hence to a white-noise spectrum. Its amplitude is more than two orders of magnitude larger than that of the differential $\mu_+ - \mu_-$ branch, whose PSD decreases approximately with frequency and remains close to the measurement floor. The output stochasticity is therefore dominated by the injected noise in the σ branch, while temporal fluctuations associated with the stored mean are negligible. The combination of a programmable memristor conductance and an external white-noise source consequently provides the two properties that intrinsic memristor variability cannot simultaneously deliver: a reproducible control of the stochastic amplitude and an approximately white temporal spectrum.

This architecture thus preserves the non-volatile and compact representation of the distribution parameters in the memristor array while moving the generation of temporal randomness to a controlled external source. The programmed conductance G_σ determines the amplitude of the injected fluctuations, rather than relying on the uncontrolled intrinsic noise of an individual device, providing a direct and reproducible route to the σ parameter required for Bayesian temporal sampling.

External noise injection solves the problem of generating controllable temporal stochasticity. It does not, however, remove device-to-device variability in the memristors that encode μ and σ. We next examine how this variability propagates into the

probability distributions represented by the hardware.

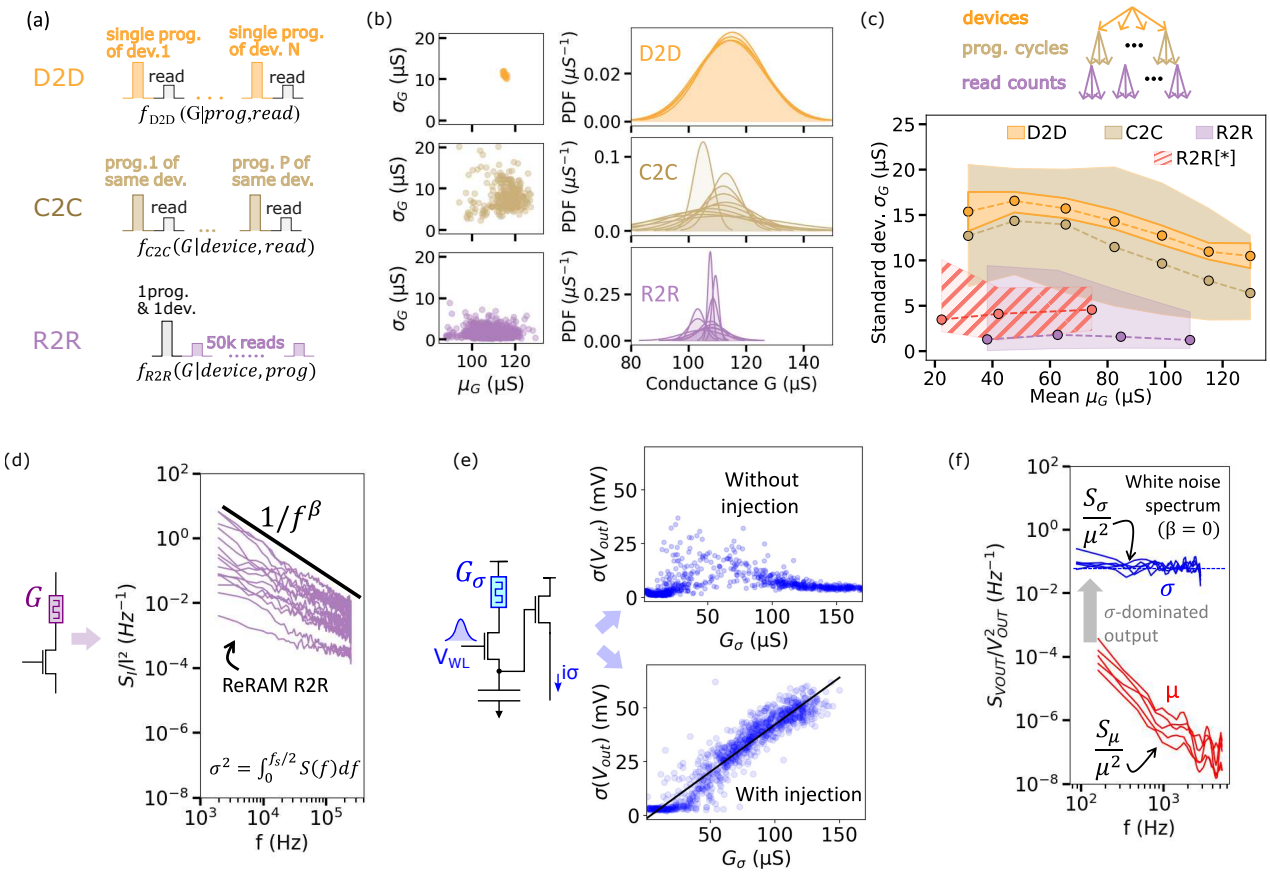


Figure 2. Intrinsic memristor variability and externally injected white noise. a Measurement protocols used to isolate device-to-device (D2D), cycle-to-cycle (C2C), and read-to-read (R2R) variability. **b** Conductance statistics extracted for the three variability sources at comparable programming targets. Each distribution is characterized by its mean μ_G and standard deviation σ_G, with the corresponding probability density functions shown for representative measurements. **c** Standard deviation σ_G as a function of mean conductance μ_G for D2D, C2C and R2R variability. Shaded regions indicate the 95% confidence intervals. R2R variability is substantially smaller than D2D variability but exhibits a large device-to-device dispersion and remains approximately independent of the programmed conductance. **d** Normalized power spectral density (PSD) of R2R fluctuations measured at $G = 130$ μS from 10,000 consecutive reads. The spectra follow an approximate $1/f^\beta$ dependence with $\beta \approx 1$, characteristic of flicker-like rather than white noise. **e** Standard deviation of 512 temporal samples from the σ branch as a function of the programmed conductance G_σ, with and without external white-noise injection. Noise injection establishes an approximately affine relationship between G_σ and $\sigma(V_{out})$, enabling programmable control of the stochastic amplitude. **f** Normalized output-noise PSD of the σ branch and of the differential $\mu_+ - \mu_-$ branch. The σ branch exhibits an approximately white spectrum ($\beta = 0$) and a noise power more than two orders of magnitude larger than that of the mean branch, demonstrating that the temporal output fluctuations are dominated by the injected noise.

Hardware-aware learning

We next quantified how programming variability translates into uncertainty in the Gaussian parameters represented by the fabricated circuit. Electrical measurements performed across 1024 $\mu_\pm$ branches and 1024 σ branches of the chip (Fig. 3a,b) establish experimentally the transfer functions used to map memristor conductances onto the mean and standard deviation of the sampled distributions. The cumulative distribution functions (CDFs) in Fig. 3a show a systematic shift of the extracted σ as G_σ is increased, demonstrating a monotonic and approximately linear control of the stochastic amplitude. This behaviour is consistent with the operation of the σ branch, in which the programmed conductance controls the effective filtering of the externally injected white noise. Similarly, the CDFs of μ_+ and μ_- in Fig. 3b shift systematically with $G_{\mu\pm}$, providing a

controlled encoding of the positive and negative contributions to the mean. These measurements establish the experimentally accessible conductance-to-parameter mappings that are subsequently used during hardware-aware training.

The finite variability of the three memristor branches, however, prevents a programmed conductance triplet $(G_\sigma, G_{\mu+}, G_{\mu-})$ from corresponding to a unique pair (μ, σ). Instead, device-to-device (D2D) variability transforms a nominal programming point into a distribution of realised Gaussian parameters. This hierarchy is illustrated in Fig. 3c: a single point in conductance space maps onto a finite region in parameter space, from which repeated temporal samples are drawn according to $V_{out} \sim \mathcal{N}(\mu, \sigma^2)$. Different programming scenarios therefore generate overlapping distributions in the $\mu - \sigma$ plane rather than deterministic parameter values. The uncertainty is particularly relevant because the three branches contribute independently to the realised parameters and consequently combine to determine the joint distribution of μ and σ.

We capture this experimentally observed uncertainty using a hierarchical probabilistic model. The independent branch statistics are described as

$$\mu^+ \sim \mathcal{N}\left(\mu^+_{\text{ideal}}, \Delta\mu^2_+\right), \qquad \mu^- \sim \mathcal{N}\left(\mu^-_{\text{ideal}}, \Delta\mu^2_-\right), \qquad \sigma \sim \mathcal{N}\left(\sigma_{\text{ideal}}, \Delta\sigma^2\right), \tag{2}$$

where the experimentally measured spreads of the individual branches are propagated to the encoded Gaussian parameters. Because the mean is obtained from the differential combination $\mu = \mu^+ - \mu^-$, its variance is given by $\Delta\mu^2 = \Delta\mu^2_+ + \Delta\mu^2_-$ for independent branches. The resulting joint distribution of the encoded parameters can therefore be represented by a bivariate Gaussian,

$$\boldsymbol{\mathcal{E}}_Q \sim \mathcal{N}_2\left(\mathbf{M}, \Sigma\right), \qquad \mathbf{M} = \begin{pmatrix} \mu_{\text{ideal}} \\ \sigma_{\text{ideal}} \end{pmatrix}, \qquad \Sigma = \begin{pmatrix} \Delta\mu^2 & 0 \\ 0 & \Delta\sigma^2 \end{pmatrix}, \tag{3}$$

with $\mu_{\text{ideal}} = \mu^+_{\text{ideal}} - \mu^-_{\text{ideal}}$. Representative measurements in Fig. 3d show that the resulting distributions form approximately elliptical confidence contours around the ideal programming point. The marginal distributions along the μ and σ axes directly quantify the experimentally observed uncertainty in each encoded parameter, while the corresponding quantile contours provide a compact representation of the probability of obtaining a given realised weight.

This formulation defines the set of experimentally realizable weights as a discrete codebook of stochastic elliptical states rather than as a continuous set of ideal Gaussian parameters. Figure 3e summarizes this mapping from the independently characterized branch distributions to the equivalent bivariate elliptical model. Each conductance combination therefore corresponds to an experimentally calibrated elliptical state $\boldsymbol{\mathcal{E}}_Q$, whose centre is set by the nominal encoded (μ, σ) and whose extent is determined by the measured D2D variability. The resulting representation explicitly incorporates both parameter quantization and device-induced uncertainty into the hardware model. The relevance of the bivariate Gaussian model is assessed in Supplementary Note 2 by comparing its predicted confidence regions with experimentally extracted density contours across a representative subset of quantized weight states $\boldsymbol{\mathcal{E}}_Q$.

To train an inference network directly against these experimentally realizable states, we adapted Hierarchical Variational Models (HVM)[19]. HVM provides a natural framework for representing uncertainty through hierarchical latent variables, which we use here to account for the stochasticity associated with the memristor-encoded elliptical states. A direct optimization of the hardware states would nevertheless be problematic because the available set $\boldsymbol{\mathcal{E}}_Q$ is discrete. We therefore introduce a hidden continuous elliptical weight $\boldsymbol{\mathcal{E}}_H$, which serves as the optimization variable while the forward model remains constrained to experimentally realizable states.

The resulting hardware-aware optimization procedure is illustrated in Fig. 3f. At each learning step k, the continuous hidden distribution $\boldsymbol{\mathcal{E}}_H^{(k)}$ is projected onto the closest experimentally realizable quantized state $\boldsymbol{\mathcal{E}}_Q$, using the Hellinger distance as the projection metric. The Bayesian loss $\mathscr{L}(\boldsymbol{\mathcal{E}}_Q)$ and its gradient are then evaluated using this quantized state, ensuring that both the forward and backward passes reflect the statistics of the actual hardware representation. The resulting gradient is used to

update the hidden continuous state according to

$$\boldsymbol{\varepsilon}_H^{(k+1)} \leftarrow \boldsymbol{\varepsilon}_H^{(k)} - l_r \left. \frac{\partial \mathscr{L}}{\partial \boldsymbol{\varepsilon}} \right|_{\boldsymbol{\varepsilon}=\boldsymbol{\varepsilon}_Q}, \tag{4}$$

where l_r is the learning rate. The updated hidden distribution is again projected onto the available codebook at the next iteration, resulting in alternating continuous gradient accumulation and discrete hardware-constrained evaluation.

This hidden-weight formulation allows the optimization to evolve in a continuous parameter space while preserving the discrete and stochastic nature of the fabricated memristor states during inference. Rather than compensating for device variability after training, the method incorporates the measured distributions of (μ, σ) directly into the optimization. The resulting learning procedure is therefore hardware-aware by construction: the network is trained using the same quantized elliptical states and uncertainty statistics that constrain the physical inference chip.

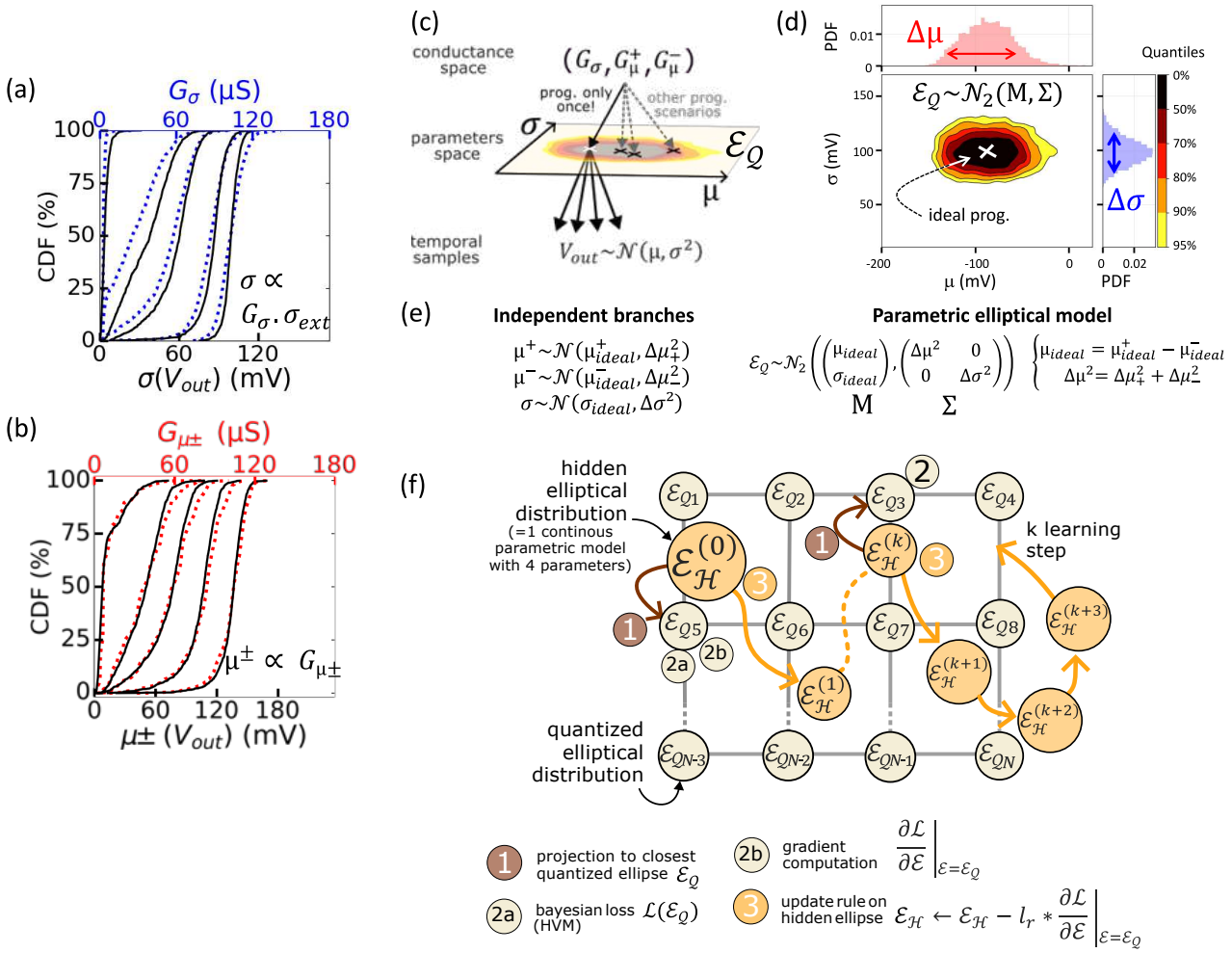


Figure 3. Hardware-aware encoding and learning with experimentally characterized memristor states. a Experimental encoding of the standard deviation σ across 1024 σ branches. Cumulative distribution functions (CDFs) of the extracted σ values (black) shift systematically with the programmed conductance G_σ (blue), demonstrating programmable control of the stochastic amplitude. **b** Experimental encoding of the differential mean across 1024 $\mu_\pm$ branches. CDFs of the extracted $\mu_\pm$ values (black) shift with the programmed conductances $G_{\mu\pm}$ (red), establishing controlled encoding of the mean contributions. **c** Hierarchical mapping from programmed conductances $(G_\sigma, G_{\mu+}, G_{\mu-})$ to the encoded distribution parameters (μ, σ) and, subsequently, to temporal samples $V_{out} \sim \mathcal{N}(\mu, \sigma^2)$. D2D variability in the three memristor branches transforms each nominal programming point into a distribution of realised Gaussian parameters. **d** Representative experimentally obtained joint distribution of (μ, σ) around the ideal programming point. Marginal distributions quantify the spreads $\Delta\mu$ and $\Delta\sigma$, while the quantile contours define the corresponding stochastic elliptical state $\boldsymbol{\mathcal{E}}_Q$. **e** Parametric elliptical model obtained by propagating the independently measured branch distributions. The encoded mean and standard-deviation uncertainties are represented by a bivariate Gaussian with mean vector **M** and covariance matrix Σ, defining the experimentally realizable elliptical states. **f** Hardware-aware training using Hierarchical Variational Models (HVM). A continuous hidden elliptical distribution $\boldsymbol{\mathcal{E}}_H$ is projected onto the closest quantized hardware state $\boldsymbol{\mathcal{E}}_Q$ at each learning step. The Bayesian loss and gradient are evaluated on $\boldsymbol{\mathcal{E}}_Q$, while the gradient is accumulated in $\boldsymbol{\mathcal{E}}_H$ before the next projection, enabling optimization despite the discrete hardware codebook.

Classification results based on training over quantized elliptical states

We next evaluated the impact of the experimentally calibrated elliptical-state representation at the network level. Training, inference and temporal sampling were performed in software on the MIT-BIH Arrhythmia dataset[20], using the experimentally measured conductance-to-parameter mappings and the corresponding stochastic elliptical states of the fabricated memristor circuit. This software-based evaluation preserves the quantization and device-to-device variability measured experimentally, while allowing the effect of temporal sampling and hardware constraints to be evaluated independently of circuit-level readout limitations.

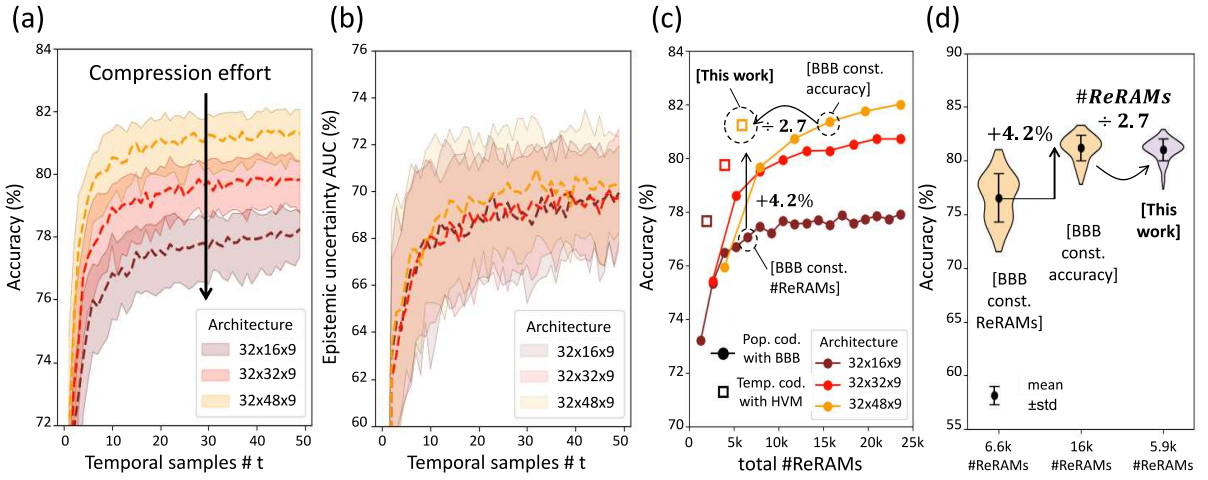
The number of temporal samples provides a direct trade-off between computational sampling and inference performance. Figure 4a shows the classification accuracy obtained for the three considered network architectures as a function of the number of temporal samples. For all architectures, accuracy increases rapidly with the number of samples and subsequently approaches a saturation regime. The same behaviour is observed for out-of-distribution (OOD) detection, evaluated using the area under the receiver-operating-characteristic curve (AUC) with epistemic uncertainty as the detection score[8] (Fig. 4b). Thus, only a finite number of temporal samples is required to recover most of the classification and uncertainty-estimation performance, providing a controllable accuracy–sampling trade-off. Further details on the confidence intervals and simulation runs are provided in Supplementary Note 4.

The temporal representation also changes the relationship between Bayesian model capacity and non-volatile memory usage. Figures 4c,d compare the simulated accuracy and memristor count of the proposed HVM-based temporal implementation with population coding and standard Bayes by Backpropagation (BBB)[1]. At comparable classification accuracy, temporal coding reduces the required number of memristors by a factor of 2.7. Conversely, when the total memristor budget is kept comparable, the temporal representation can accommodate a larger network and increases the classification accuracy by 4.2 percentage points. These two operating points illustrate complementary benefits of the proposed representation: temporal coding can either reduce the non-volatile memory required for a given accuracy or increase model capacity for a fixed hardware budget.

The origin of this compression is that the Gaussian weight distribution is represented by three memristors per synaptic weight, independently of the number of temporal samples used during inference. The additional samples therefore increase the number of stochastic evaluations of the same stored distribution rather than requiring additional non-volatile storage. In contrast, the number of stored parameters in the proposed implementation is fixed by the three-device representation of (μ_+, μ_-, σ). Temporal sampling consequently provides an additional computational dimension without increasing the memristor count. The resulting separation between memory footprint and sampling resolution allows classification accuracy and epistemic uncertainty estimation to be improved through temporal averaging without a proportional increase in non-volatile memory.

Figure 4e places the proposed implementation in the context of previous Bayesian hardware approaches based on population, write-based and read-based temporal coding. The present approach combines analog encoding of the mean and standard deviation with analog generation of white-noise fluctuations in HfOx:Si memristors integrated in a 130 nm technology. The experimentally calibrated elliptical-state representation further incorporates device variability into the training procedure, rather than treating the programmed conductances as ideal deterministic parameters. Together with the three-memristor-per-weight representation, this provides a compact non-volatile implementation in which stochastic sampling is generated temporally and its amplitude is controlled through the programmed σ branch.

Overall, these results show that the proposed representation retains the benefits of Bayesian inference and temporal uncertainty estimation while decoupling the number of temporal samples from the non-volatile memory footprint. The experimentally constrained HVM therefore provides a route to inference models that are simultaneously aware of memristor quantization, device variability and the finite sampling budget of the hardware.



(e)

		Population coding	Write-based temporal coding		Read–based temporal coding		
Ref.		Bonnet, Nat. Commun. 2023	You, ISSCC, 2025	Sebastian, Nat Commun. 2022	Lin, IEDM 2019	Shan, IEDM 2024	**This work**
Technology		130nm HfO_x:Si memristor or GST PCM	22 nm STT-MRAM	MoS2 2D memtransistor	TaOx-HfOx memristor	40nm TaOx memristor + VO_2	130nm HfO_x:Si memristor
Algorithm		Bayes by backprop	Bayes by backprop	Bayes by backprop	Bayes by backprop	Bayes by backprop	Hierachical Variational Model
Weight encoding	**μ**	Analog D2D conductance sample	Digital μ	Analog C2C conductance sample	Analog R2R conductance samples over several devices	Analog mu	Analog mu
	σ		Analog C2C conductance sample			Analog sigma scaling	Analog sigma scaling
	Noise source					RTN	Analog white noise
Footprint		X	✓	✓	~	✓	✓
D2D & C2C robustness		✓	✓	✓	X	X	✓
Cycle endurance		✓	X	X	✓	✓	✓
Energy		✓	X	X	✓	✓	✓
Non-volatile		✓	X	X	✓	✓	✓

Figure 4. Inference performance and hardware trade-offs of temporal coding with quantized elliptical states. a,b Classification accuracy and epistemic-uncertainty AUC for out-of-distribution detection as a function of the number of temporal samples for the three considered network architectures. Both metrics increase rapidly with the number of samples before approaching a saturation regime. **c** Classification accuracy as a function of the total number of memristors (ReRAMs) for the proposed HVM-based temporal implementation and Bayes by Backpropagation (BBB). The two marked operating points correspond to either comparable accuracy, for which temporal coding reduces the memristor count by 2.7×, or comparable memory budget, for which temporal coding provides a 4.2 percentage-point accuracy improvement. **d** Summary of the constant-accuracy and constant-memristor operating points, illustrating the trade-off between inference performance and non-volatile memory footprint. **e** Comparison with representative Bayesian hardware implementations based on population, write-based and read-based temporal coding. The proposed approach combines analog encoding of μ and σ with externally generated white-noise sampling in 130 nm HfOx memristors, using three memristors per weight independently of the number of temporal samples.

Discussion

Here, we have shown that decoupling the non-volatile storage of Bayesian weight parameters (μ, σ) from the temporal generation of stochastic samples provides a viable route to compact memristor-based Bayesian hardware. Intrinsic cycle-to-cycle and read-to-read memristor fluctuations exhibit a non-white, $1/f^{\beta}$ spectrum and an amplitude that is only weakly and unpredictably related to the programmed conductance, making them unsuitable as a direct entropy source for temporal sampling. By instead injecting an externally generated white-noise process whose amplitude is set through a dedicated σ branch, the proposed circuit recovers the reproducibility and spectral whiteness that intrinsic variability alone cannot provide, while retaining a compact, three-memristor-per-weight, non-volatile representation of each synaptic weight distribution.

A second contribution of this work is the hierarchical variational training procedure, which explicitly incorporates the experimentally measured device-to-device variability of the three memristor branches into the learning process rather than treating programmed conductances as ideal, deterministic targets. By representing each synaptic weight as a quantized elliptical state and optimizing a continuous hidden variable projected onto this hardware-constrained codebook at every training step, the network is trained directly against the statistics of the fabricated circuit. Evaluated at the network level on the MIT-BIH Arrhythmia dataset, this hardware-aware approach reduces the number of memristors required to reach a target accuracy by a factor of 2.7× relative to population coding trained with Bayes by Backpropagation, or alternatively improves accuracy by 4.2 percentage points at matched memristor count – using only three memristors per weight, independently of the number of temporal samples drawn during inference (Fig. 4c,d). Similar accuracy–memory trade-offs were obtained when applying the same hardware-calibrated training methodology to image classification tasks (MNIST and EMNIST Letters, Supplementary Note 5), indicating that the benefits of temporal coding with HVM are not specific to the ECG classification task.

These results indicate that separating spatial, non-volatile storage from externally controlled temporal stochasticity avoids both the memory overhead inherent to population coding and the endurance and write-energy penalties associated with cycle-to-cycle, write-based sampling schemes (Fig. 4e). Several aspects of the present demonstration nonetheless call for further investigation. First, temporal randomness is currently supplied by a bench-top arbitrary waveform generator; realizing the full footprint and energy benefit of the approach at the system level will require integrating a compact, on-chip Gaussian random-number generator, and quantifying its area and energy overhead against the memristor-count savings reported here. Second, the granularity of the experimentally realizable codebook of elliptical states, set by the device-to-device variability of the three memristor branches, imposes an intrinsic limit on achievable precision; characterizing this precision–compactness trade-off, and exploring whether it can be tightened through improved device uniformity or programming strategies, is an important direction for future work.

Ref.[12] also relies on an external stochastic source to generate Gaussian-distributed samples, using a differential pair of VO_2-based bitcells whose Bernoulli switching events are weighted by the conductance of paired one transistor – one memristive cells and accumulated so that a Gaussian distribution emerges through the central limit theorem. Our architecture achieves the same outcome with a simpler, single-branch implementation: rather than combining two independent stochastic sources through a differential pair, temporal charge differencing on a single memristor branch amplifies the injected noise while removing its DC component. This simplification suggests that Gaussianity need not originate from the stochastic source itself. Provided that the source generates sufficiently decorrelated, finite-variance samples, Gaussian statistics can instead emerge through temporal accumulation under the conditions of the central limit theorem. Although this mechanism is not directly tested in the present work, which relies on a stochastic source whose sample distribution has already been verified to be Gaussian (Methods), it points to a broader design space in which non-Gaussian stochastic sources may serve as entropy sources for memristor-based GRNGs.

Looking forward, combining the present temporal-coding architecture with a compact, on-chip entropy source would enable a fully integrated, non-volatile and endurance-free Bayesian inference chip. More broadly, by treating device-to-device variability as a modeled component of the hardware rather than a nuisance to be suppressed, the hierarchical variational approach introduced here offers a general strategy for deploying uncertainty-aware neural networks on memristive hardware under

realistic, non-ideal operating conditions.

Methods

Chip fabrication

The demonstrator circuit was fabricated using Leti's MAD200 technology, based on a 130 nm CMOS process with TiN/Ti/HfO_x:Si/TiN memristors integrated in the back-end-of-line (BEOL) between metal levels 4 and 5. The memristors comprise a 10 nm-thick Si-doped HfO_x switching layer between TiN electrodes and a Ti scavenging layer introduced to reduce the forming voltage. The memristor pillars were patterned with a diameter of 300 nm. The fabrication process was similar to that reported in Ref.[8].

Design of test chip

The test chip comprises a $2 \times 32 \times 32$ memristor array, corresponding to 1024 σ branches and 1024 $\mu_\pm$ branches, together with an on-chip charge trans-impedance amplifier (CTIA) implementing the charge-accumulation readout scheme. A full-custom analogue design was adopted with 5 V high-voltage CMOS devices used to interface with the memristor array. Electrical simulations were performed using Siemens Eldo, the circuit and layout were designed in Cadence Virtuoso, and layout-versus-schematic (LVS) and design-rule-check (DRC) verifications were performed using Calibre. Since the μ_+ and μ_- bitcells have identical schematics, their results are reported using the $\mu_\pm$ notation. The RC bitcells, corresponding to the σ branches, share a 10 pF capacitor between the bitlines to reduce the array footprint. The CTIA incorporates a 4 pF feedback capacitor for on-chip charge sensing, enabling the measurement of noise-induced charge fluctuations under excitation from off-chip waveform generators (WFGs).

Temporal sampling with Bayesian demonstrator

To reduce design and verification time and avoid the development of a dedicated printed circuit board, the circuit was implemented in a scribe-line test structure comprising 25 I/O pads for wafer-level characterization. The pads provide the digital control signals required for memristor addressing and CTIA initialization, as well as dedicated analogue connections for memristor programming and bitcell readout through V_{TOP}, V_{DRAIN}, V_{SOURCE}, V_{WL}, and V_{GATE}. Electrical characterization was performed using a Keysight B1500 parameter analyzer for DC biasing, a Keysight B1530 waveform generator/fast measurement unit for synchronized V_{TOP} and V_{DRAIN} waveforms, and a Keysight 33600A waveform generator to generate a band-limited Gaussian noise signal with standard deviation $\sigma = 10$ mV and bandwidth $f_B = 60$ kHz. The Gaussian statistics of the applied V_{WL} signal were verified at the probe-card level using a Teledyne LeCroy HDO4000 oscilloscope with a sampling rate of 1 MHz. Digital addressing and analogue-to-digital conversion at the end of each accumulation sequence were controlled using an Arduino Mega 2560 board.

Intrinsic variability estimation

Our approach sought to isolate the effects of programming variability from those of read variability. First, we quantified C2C and D2D variability while minimizing the contribution of read-related fluctuations. To this end, we used 70µs read pulses with a 1µs ramp, and the signal was integrated over the final 50% of the pulse duration. Seven groups of 256 memristors with increasing compliance current were programmed 19 times and read once to evaluate the amplitude of D2D and C2C. Similarly, R2R fluctuations can be analyzed while minimizing C2C and D2D variabilities with smart program-and-verify scheme : 4 independent groups of 1024 devices were encoded and a sequence 50k pulses was used to sample conductance with a 2µs period and 500ns integration time. We used Keysight B1530 to generate read pulses of 400mV and gate voltage of 5.3V.

Hierarchical Variational Model

HVM builds on the same Bayesian variational-inference framework used to approximate the posterior distribution of the model parameters. Given a dataset $\mathscr{D}$, the posterior distribution over the network weights is denoted by $p(z|\mathscr{D})$, where z represents the set of network weights. In a Bayesian neural network (BNN), this posterior is approximated by a variational distribution $q(z|\theta)$

parameterized by θ. In the conventional formulation, θ consists of the set of mean and standard-deviation parameters of the BNN weights.

Using Bayes' rule, the variational objective can be expressed through the negative evidence lower bound (ELBO)[21,22]:

$$\mathscr{L} = \underset{z\sim q_\theta}{\mathbb{E}}\left(log(\frac{q_\theta(z)}{p(z).p(\mathscr{D}|z)})\right) = \underset{z\sim q_\theta}{\mathbb{E}}(logq_\theta(z)) - \underset{z\sim q_\theta}{\mathbb{E}}(logp(z)) - \underset{z\sim q_\theta}{\mathbb{E}}(logp(\mathscr{D}|z)) \tag{5}$$

The first two terms account for the divergence between the variational posterior and the prior, whereas the last term corresponds to the expected negative log-likelihood of the observed data. The expectations are evaluated using Monte Carlo sampling, with weight samples drawn from the variational distribution, as in Bayes by Backprop (BBB)[1].

In the proposed HVM formulation, each BNN weight is associated with an elliptical state $\mathcal{E}$ in the (μ,σ) parameter space (see Fig. 3). The Gaussian weight distribution is therefore obtained hierarchically by first sampling the parameters (μ,σ) from the elliptical distribution and then sampling the weight z from the corresponding Gaussian distribution. Under a mean-field approximation, in which the weights are assumed to be independent, the variational posterior can be written as

$$q_\theta(z) = \prod_{\mathcal{E}}\iint_{(\mu,\sigma)} f_{\mu,\sigma}(z) f_{\mathcal{E}}(\mu,\sigma)\,d\mu\,d\sigma \text{ where } \begin{cases} \theta = \{\text{ensemble of elliptical weight } \varepsilon \text{ of BNN}\} \\ f_{\mu,\sigma} \sim \mathscr{N}(\mu,\sigma^2) \\ f_{\mathcal{E}} \sim \mathscr{N}_2(M,\Sigma) \end{cases} \tag{6}$$

Here, $f_{\mu,\sigma}(z)$ describes the conditional Gaussian distribution of an individual network weight for a given pair (μ,σ), while $f_{\mathcal{E}}(\mu,\sigma)$ describes the distribution of the Gaussian parameters induced by the experimentally characterized hardware variability. The latter is represented by a bivariate Gaussian with mean vector $\mathbf{M}$ and covariance matrix Σ, corresponding to the elliptical states introduced in Fig. 3. This hierarchical formulation therefore propagates uncertainty from the hardware-level parameters (μ,σ) to the distribution of individual BNN weights.

For a given minibatch, weight samples are drawn hierarchically by first sampling (μ,σ) from the elliptical state $\mathcal{E}$ and then sampling z from $\mathscr{N}(\mu,\sigma^2)$. The sampled weights are used to perform the forward pass and compute the neural-network output. The expected log-likelihood is subsequently estimated from these stochastic network evaluations and optimized using minibatch stochastic gradient descent (SGD)[22]. This procedure allows the uncertainty associated with the experimentally realizable elliptical states to be propagated throughout both training and inference.

Acknowledgements

This work was supported by the European Research Council (ERC Consolidator Grant DIVERSE, 101043854) and the France 2030 program (ANR, BEP: ANR-22-PEEL-0010). Large language models were used to support the linguistic revision of selected sections of this manuscript.

Author contributions

T.J. designed the single bitcell with Eldo simulations under supervision of O.B. and J.M.P.. Layout and PEX simulations of demonstrator including 2k array and peripheral circuitry was conducted by O.B.. Electrical tests of fabricated chip were conducted by T.J.. HVM framework was simulated by homemade python package implemented by T.J.. Quantized HVM model refinement was widely discussed with T.H. and D.B.. J.MP. and E.V. supervised the work. All authors discussed the results and contributed to the final manuscript.

Competing interests

Damien Querlioz is a Guest Editor of the Collection Future Computing: Paradigms and Design for Next Generation Applications. Damien Querlioz was not involved in the journal's review of, or decisions related to, this paper. The remaining authors declare no competing interests.

Data availability

The MIT BIH dataset is available online[20]; additional details regarding ECG data preparation are available in Supplementary Note 3. The MNIST and EMNIST Letters datasets used in this study were obtained through the `torchvision.datasets` package. The bitcell experimental data generated from Bayesian demonstrator are available from the corresponding author upon reasonable request. The experimental data to simulate population coding with BBB was taken from Ref.[8].

Code availability

The software developed for this study is not publicly available but may be shared upon reasonable request to the corresponding author.

References

1. Blundell, C., Cornebise, J., Kavukcuoglu, K. & Wierstra, D. Weight Uncertainty in Neural Networks, DOI: 10.48550/arXiv.1505.05424 (2015). ArXiv:1505.05424 [stat.ML].
2. Blei, D. M., Kucukelbir, A. & McAuliffe, J. D. Variational inference: A review for statisticians. *J. Am. statistical Assoc.* **112**, 859–877 (2017).
3. Jospin, L. V., Laga, H., Boussaid, F., Buntine, W. & Bennamoun, M. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Comput. Intell. Mag.* **17**, 29–48 (2022).
4. Querlioz, D. & Vianello, E. Bayesian electronics for trustworthy artificial intelligence. *Nat. Rev. Electr. Eng.* **2**, 846–855, DOI: 10.1038/s44287-025-00226-x (2025).
5. Cai, R., Ren, A., Liu, N., Ding, C., Wang, L., Qian, X., Pedram, M. & Wang, Y. Vibnn: Hardware acceleration of bayesian neural networks. *ACM SIGPLAN Notices* **53**, 476–488 (2018).
6. Dalgaty, T., Castellani, N., Turck, C., Harabi, K.-E., Querlioz, D. & Vianello, E. In situ learning using intrinsic memristor variability via markov chain monte carlo sampling. *Nat. Electron.* **4**, 151–161 (2021).
7. Jung, S., Lee, H., Myung, S., Kim, H., Yoon, S. K., Kwon, S.-W., Ju, Y., Kim, M., Yi, W., Han, S. *et al.* A crossbar array of magnetoresistive memory devices for in-memory computing. *Nature* **601**, 211–216 (2022).
8. Bonnet, D., Hirtzlin, T., Majumdar, A., Dalgaty, T., Esmanhotto, E., Meli, V., Castellani, N., Martin, S., Nodin, J.-F., Bourgeois, G., Portal, J.-M., Querlioz, D. & Vianello, E. Bringing uncertainty quantification to the extreme-edge with memristor-based Bayesian neural networks. *Nat. Commun.* **14**, 7530, DOI: 10.1038/s41467-023-43317-9 (2023).
9. Sebastian, A., Pendurthi, R., Kozhakhmetov, A., Trainor, N., Robinson, J. A., Redwing, J. M. & Das, S. Two-dimensional materials-based probabilistic synapses and reconfigurable neurons for measuring inference uncertainty using Bayesian neural networks. *Nat. Commun.* **13**, 6139, DOI: 10.1038/s41467-022-33699-7 (2022).
10. You, D.-Q., Khwa, W.-S., Zhang, B., Chen, F.-Y., Lee, A., Hung, Y.-C., Li, Y.-M., Wang, Y.-H., Lo, C.-C., Liu, R.-S. *et al.* 14.1 a 22nm 104.5 tops/w μ-nmc-δ-imc heterogeneous stt-mram cim macro for noise-tolerant bayesian neural networks. In *2025 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 68, 1–3 (IEEE, 2025).

11. Lin, Y., Zhang, Q., Tang, J., Gao, B., Li, C., Yao, P., Liu, Z., Zhu, J., Lu, J., Hu, X. S., Qian, H. & Wu, H. Bayesian Neural Network Realization by Exploiting Inherent Stochastic Characteristics of Analog RRAM. In *2019 IEEE International Electron Devices Meeting (IEDM)*, 14.6.1–14.6.4, DOI: 10.1109/IEDM19573.2019.8993616 (IEEE, San Francisco, CA, USA, 2019).

12. Shan, L., Wu, L., Wang, Z., Xie, R., Ban, C., Yang, G., Wang, Q., Li, Y., Ma, H., Bao, L., Liang, L., Wang, Y., Cai, Y. & Huang, R. Monolithically 3D Integrated Memristive Bayesian Neural Network for Intelligent Motion Planning. In *2024 IEEE International Electron Devices Meeting (IEDM)*, 1–4, DOI: 10.1109/IEDM50854.2024.10873381 (IEEE, San Francisco, CA, USA, 2024).

13. Song, M., Koo, R.-H., Kim, J., Han, C.-H., Yim, J., Ko, J., Yoo, S., Choe, D.-h., Kim, S., Shin, W. *et al.* Ferroelectric nand for efficient hardware bayesian neural networks. *Nat. Commun.* **16**, 6879 (2025).

14. Liu, G., Ding, J., Wan, Z., Xue, J., Zhang, Y., Gong, Z., Peng, C., Shi, W., Su, J., Xia, J. *et al.* Coupled dual-channel memristors for hardware-native trustworthy bayesian intelligence. *Nat. Commun.* (2026).

15. Gong, T., Wu, Q., Huang, Y., Jiang, H., Yang, J., Luo, Q., Chung, S. S. & Liu, M. First Demonstration of a Bayesian Machine based on Unified Memory and Random Source Achieved by 16-layer Stacking 3D Fe-Diode with High Noise Density and High Area Efficiency. In *2023 International Electron Devices Meeting (IEDM)*, 1–4, DOI: 10.1109/IEDM45741.2023.10413799 (IEEE, San Francisco, CA, USA, 2023).

16. Huang, Y., Wu, Q., Gong, T., Yang, J., Luo, Q. & Liu, M. Bayesian neural network with unified entropy source and synapse weights using 3d 16-layer fe-diode array. *Nat. Commun.* **16**, 8063 (2025).

17. Lin, Y., Zhang, Q., Gao, B., Tang, J., Yao, P., Li, C., Huang, S., Liu, Z., Zhou, Y., Liu, Y. *et al.* Uncertainty quantification via a memristor bayesian deep neural network for risk-sensitive reinforcement learning. *Nat. Mach. Intell.* **5**, 714–723 (2023).

18. Puglisi, F. M., Zagni, N., Larcher, L. & Pavan, P. A new verilog-a compact model of random telegraph noise in oxide-based rram for advanced circuit design. In *2017 47th European Solid-State Device Research Conference (ESSDERC)*, 204–207 (IEEE, 2017).

19. Ranganath, R., Tran, D. & Blei, D. Hierarchical variational models. In *International conference on machine learning*, 324–333 (PMLR, 2016).

20. Moody, G. & Mark, R. The impact of the MIT-BIH Arrhythmia Database. *IEEE Eng. Medicine Biol. Mag.* **20**, 45–50, DOI: 10.1109/51.932724 (2001).

21. Awano, H. & Hashimoto, M. B2n2: Resource efficient bayesian neural network accelerator using bernoulli sampler on fpga. *Integration* **89**, 1–8 (2023).

22. Diederik, P. K. & Max, W. An introduction to variational autoencoders. *Foundations Trends Mach. Learn.* **12**, 307–392 (2019).

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SupplementaryinformationICRCTemporalcoding.zip