# A Per-Chip Metadata Correction Framework for Neuromorphic Memristor Crossbars: Theoretical Architecture and Viability Analysis

Naman Boggaram

## 1. Introduction

**Scope and theoretical basis.** The analyses presented in this document are theoretical, derived from published device parameters of Goswami et al. [1]. Experimental validation of per-chip characterisation metadata generation, measurement noise bounds, and long-term drift behaviour under the heartbeat protocol requires chip-level access not available at the time of writing. This work establishes the theoretical framework and viability conditions; experimental confirmation is identified as future work.

The first question I asked when encountering variability in memristor crossbar arrays was whether it was actually a problem in the way the field treats it.

In binary transistor-based systems, variability is catastrophic. A transistor that switches at the wrong threshold turns a 1 into a 0. The entire computation fails. The precision requirement is absolute because the representation is discrete there is no such thing as a "slightly wrong" binary value.

A memristor crossbar is not a binary system. It is an analog one. The computation it performs matrix-vector multiplication through Ohm's law at each crosspoint and Kirchhoff's current summation at each column is inherently proportional. If a node's conductance lands at $G_{\text{ideal}} + \delta$ instead of $G_{\text{ideal}}$, the output current is proportionally offset. The error is not catastrophic. It is continuous, bounded, and structured.

This observation led to a different question: not *how do we eliminate the variability*, but *can we compute for it*.

The immediate difficulty is that computing for variability on a per-output basis would require knowing the full input at inference time before the inference runs a circular dependency. Per-output correction is not tractable.

But the interface layer between the model and the hardware operates at a different moment: model loading, not inference. When a pre-trained model's weight matrix $W$ is being written into the crossbar's conductance states, the correction can be applied once, completely, before any inference call is made. If the per-node offset $\delta(i, j)$ is known for every node from a prior characterisation measurement, the programming target can be pre-compensated:

$$W_{\text{corrected}}(i,j) \;=\; \frac{W(i,j) \;-\; \delta(i,j)}{\alpha(i,j)} \tag{1}$$

where $\alpha(i,j)$ is the per-node gain coefficient. The device's physical offset then cancels the pre-compensation exactly, and the crossbar implements $W$ as intended. This correction is computed once per model load a single matrix operation taking microseconds on any embedded processor and carries zero overhead during inference itself.

Three properties follow from this architecture that I did not initially anticipate but which make it practically significant.

**First**, the correction is exact rather than statistical under the measurability condition $\sigma_{\text{meas}} < \Delta G_{\min}/6$ — a condition satisfied by the molecular memristor devices of Goswami et al. [1] as shown in Figure 8e of that work, subject to the linearity assumptions discussed in Section 4. Hardware-aware training [2] reduces the *expected* error across a population of devices. Per-chip correction eliminates the error for this specific device under this condition. Direct experimental verification of this condition on production-grade chips remains an open empirical question.

**Second**, the approach decouples model development from hardware development. The same pre-trained model whether a large language model, a narrow defence classification network, or any architecture expressible as matrix multiplications can be deployed on any characterised crossbar chip regardless of its specific variability profile. The correction absorbs hardware-specific offsets without modifying the model. This enables rapid switching between models and between successive versions of the same model on a single blank chip: load new weights, apply the stored correction, begin inference.

**Third**, the correction remains accurate over device lifetime through a periodic recalibration protocol a "heartbeat" that re-measures current conductance states, computes updated offsets, and reprograms only the nodes whose drift has exceeded a defined threshold. From the stability data in Goswami et al., a heartbeat interval of one hour consumes 14.78 mJ and 47 ms per cycle for a $4000 \times 4000$ array, representing 0.00058% of operational time.

The prerequisite for all of this is the per-node characterisation data: the measured $\alpha(i,j)$ and $\delta(i,j)$ for every node in the array. This data does not currently exist as a systematic manufacturing output for any memristor device class. Producing it requires an automated characterisation instrument that sweeps every node through its full conductance range, fits a linear response model to each, and generates a standardised metadata file that travels with the chip from fabrication to deployment.

That instrument, and the question of whether the variability field in molecularly engineered memristors has sufficient spatial structure to be compactly represented, is what this proposal addresses.

The remainder of this document is structured as follows. Section 2 compares this approach with hardware-aware training. Section 3 analyses energy requirements. Section 4 states the honest

limitations. Section 5 provides the full computational viability analysis.

## 2. Comparison with Hardware-Aware Training

Hardware-aware training (HAT) has been established as a natural baseline for deploying neural networks on analog hardware with device variability [2, 4]. The comparison between the two approaches is not one of right and wrong but of what each requires, what each achieves, and what each gives up.

### 2.1 What hardware-aware training does

HAT couples the trained model to the hardware. The training process runs with knowledge of the device's variability profile through noise injection modelling the statistical distribution $P(\delta)$, or through direct on-hardware training. The trained weights $W_{\mathrm{HAT}}$ satisfy:

$$W_{\mathrm{HAT}} \;=\; \arg\min_{W} \mathcal{L}_{\mathrm{task}}(W) \;+\; \lambda \cdot \mathbb{E}_{\delta \sim P(\delta)} \left[ \| f(X,\, W+\delta) - f(X,\, W) \|^2 \right] \tag{2}$$

where $\lambda$ is the robustness regularisation weight and $f$ is the network function. The second term penalises sensitivity to device offsets, forcing the network to learn representations that degrade gracefully under expected variability. This is mathematically a regularisation term: it reduces model capacity in exchange for hardware robustness.

The residual per-chip inference error after HAT is:

$$\varepsilon_{\mathrm{HAT}}(i,j) \;=\; \delta_{\mathrm{actual}}(i,j) \;-\; \mathbb{E}[\delta(i,j)] \tag{3}$$

This residual is nonzero for every physical chip because every chip's actual offset deviates from the population mean. HAT corrects for the average. It cannot correct for the specific.

### 2.2 What per-chip correction does differently

The present method imposes no constraint on how the model was trained. The correction in Equation (1) operates entirely at the interface layer. The residual inference error is:

$$\varepsilon_{\mathrm{corrected}}(i,j) \;\approx\; 0 \text{ [S]} \tag{4}$$

under the measurability condition $\sigma_{\mathrm{meas}} < \Delta G_{\mathrm{min}}/6$, which is satisfied by the published characterisation data of Goswami et al. [1] [Fig. 8e] subject to the linearity assumptions of Section 4. Direct experimental verification of this condition on production-grade chips remains an open empirical question.

### 2.3 The non-compensated model advantage

**Measurability condition derivation.** The condition $\sigma_{\text{meas}} < \Delta G_{\min}/6$ is a 6-sigma argument: if measurement noise on each characterisation reading is $\sigma_{\text{meas}}$, the stored offset $\delta(i,j)$ carries an error $\varepsilon \sim \mathcal{N}(0,\, \sigma^2_{\text{meas}})$. For the correction to reliably distinguish the nearest two conductance levels (separated by $\Delta G_{\min}$), the probability of misclassification must be negligible. Requiring $6\sigma_{\text{meas}} < \Delta G_{\min}$ bounds this probability at $< 10^{-9}$ per node, yielding:

$$\sigma_{\text{meas}} \;<\; \frac{\Delta G_{\min}}{6} \tag{5}$$

From the device data in Goswami et al. [1] [Fig. 8e], this condition is satisfied under the reported characterisation protocol.

This is the property that HAT structurally cannot replicate: a model that has never seen a memristor crossbar can be loaded and run accurately.

HAT requires that the model be retrained whenever it is deployed on a new device class, a new process node, or hardware whose variability profile differs from the training assumption. Every existing pre-trained model GPT variants, vision transformers, diffusion models, any model trained on GPU clusters without neuromorphic hardware knowledge would require retraining or fine-tuning under hardware-aware objectives before deployment on memristor hardware.

Per-chip correction deploys any such model directly and without modification. The existing ecosystem of pre-trained models becomes immediately deployable on characterised crossbar chips. The correction absorbs the hardware's variability at load time; the model never needs to know the hardware exists.

Formally, let $W^*$ denote the optimal weights for the task on ideal hardware. HAT produces $W_{\text{HAT}} \neq W^*$. The gap $W_{\text{HAT}} - W^*$ represents capacity lost to robustness regularisation. Per-chip correction deploys $W^*$ directly. On ideal hardware, $W^*$ outperforms $W_{\text{HAT}}$. On characterised crossbar hardware under the present method, $W^*$ also achieves near-exact inference at the moment of load.

Under freshly applied characterisation data and the linearity assumption, the present method achieves lower residual inference error than HAT by construction. This advantage narrows between heartbeat cycles as device drift accumulates: gradual drift reduces the accuracy of the stored $\delta(i,j)$ values, and the effective residual error grows toward the HAT baseline until the next heartbeat recalibration.

1

[1]The HAT energy comparison in Section 3 assumes a single-chip deployment context. In large-scale production deployments where a single HAT-trained model is deployed across $N$ chips, the per-chip training energy scales as $E_{\text{HAT}}/N$, improving HAT's energy position. Per-chip correction's heartbeat cost is invariant to $N$.

## 3. Energy Requirements

The following analysis uses parameters derived from Goswami et al. [1] and standard embedded hardware specifications. Array size: $N = 4000$, total nodes $= 16 \times 10^6$. Write pulse energy: 252 pJ (from Figure 8d: $0.9\,\text{V} \times 3.5\,\text{mA} \times 80\,\text{ns}$). Read energy per node: 25 pJ. Processor: ARM Cortex A53 at 500 mW, 1.2 GHz. Heartbeat interval: 3,600 s. Drift fraction per cycle: 1%. Deployment life: 10 years (87,600 heartbeat cycles; 520 model loads).

### 3.1 Per-chip correction energy

| **Operation** | **Energy per event** | **Events / 10 yr** |
|---|---|---|
| Characterisation sweep (factory) | 1.35 mJ | 1 |
| Model load: dehashing | 0.156 mJ | 520 |
| Model load: compensation compute | 13.3 mJ | 520 |
| Model load: crossbar programming | 40.32 mJ | 520 |
| Heartbeat: full cycle | 14.78 mJ | 87,600 |
| Per-inference overhead | 0 mJ | $315 \times 10^9$ |

Table 1: Energy costs for per-chip metadata correction, $4000 \times 4000$ array, Architecture C full per-node storage (312 KB). The heartbeat figure of 14.78 mJ per cycle corresponds to Architecture C; see Table  for Architecture A/B values.

Total 10-year energy:

$$\begin{aligned} E_{\text{correction}} &= E_{\text{char}} + N_{\text{loads}}\,(E_{\text{dehash}} + E_{\text{comp}} + E_{\text{prog}}) + N_{\text{hb}}\,E_{\text{hb}} \\ &= 1.35\,\text{mJ} + 520 \times 53.78\,\text{mJ} + 87{,}600 \times 14.78\,\text{mJ} \\ &\approx \mathbf{1{,}322.7}\,\text{J} \end{aligned} \tag{6}$$

### 3.2 Hardware-aware training energy

HAT's primary cost is in training, not deployment. Conservative estimate for a narrow inference network: $10^6$ operations per training iteration, $10^4$ iterations, GPU energy $10^{-9}$ J per operation:

$$E_{\text{HAT, one run}} \;=\; 10^6 \times 10^4 \times 10^{-9} \;=\; 10\,\text{J} \tag{7}$$

Note that 10 J per run is a deliberately conservative lower bound; real GPU training of non-trivial networks typically consumes hundreds to thousands of joules. Even this lower bound (10 J) lies below the crossover threshold derived in Section 3.3 (32.5 J), which means the crossover argument holds under any realistic training cost.

At quarterly retraining over 10 years (40 runs), training energy alone is 400 J. Crossbar programming at deployment adds $520 \times 40.32\,\text{mJ} = 20.97$ J. Total HAT energy: approximately $\mathbf{420.97}$ J.

| Energy component | Per-chip correction | HAT |
|---|---|---|
| Hardware characterisation (one-time, per chip) | 0.00135 J | |
| Model training (recurring, per retraining run) | | 400 J (conservative) |
| Deployment programming | 27.97 J | 20.97 J |
| Heartbeat / recalibration | 1,294.7 J | 0 J |
| Per-inference overhead | 0 J | 0 J |
| **Total (10 yr)** | **1,322.7 J** | **≈ 421 J** |

Table 2: Ten-year energy comparison. HAT training estimate is conservative; real GPU training costs are typically orders of magnitude higher. The per-chip correction total uses Architecture C heartbeat figures. See footnote in Section 2.3 for the multi-chip deployment scaling argument.

### 3.3 Comparison and crossover analysis

The crossover point at which both methods consume equal total energy over 10 years is found by setting the two totals equal. Denoting the HAT deployment programming cost as $E_{\text{prog,HAT}} = 520 \times 40.32\,\text{mJ} = 20.97$ J:

$$40 \cdot E_{\text{HAT, one run}} + E_{\text{prog,HAT}} \;=\; E_{\text{correction}} \tag{8}$$

$$40 \cdot E_{\text{HAT, one run}} \;=\; E_{\text{correction}} - E_{\text{prog,HAT}} \;=\; 1{,}322.7 - 20.97 \;=\; 1{,}301.7\,\text{J} \tag{9}$$

$$\boxed{E_{\text{HAT, one run, crossover}} \;=\; \frac{1{,}301.7}{40} \;\approx\; 32.5\,\text{J per training run}} \tag{10}$$

Below 32.5 J per training run, HAT is more energy-efficient over a 10-year deployment. Above it, per-chip correction is cheaper. Real training runs for non-trivial networks on GPU clusters consume hundreds to thousands of joules, placing practical deployments well above this crossover. Furthermore, if the heartbeat interval is extended from 1 hour to 6 hours justified if operational drift data confirm the multi-month stability reported in Goswami et al. heartbeat energy falls to 215.8 J and total 10-year energy to 244.8 J, below HAT at any training cost above 5.6 J per run.

## 4. Honest Limitations

Three limitations of the present approach must be stated directly.

**The characterisation infrastructure prerequisite.** Per-chip correction requires the automated characterisation instrument to exist and be integrated into the manufacturing process before any chip can be corrected. HAT requires only a training pipeline, which already exists for any GPU-trained model. Until the characterisation instrument exists, HAT is the only viable approach for devices without per-chip measurement infrastructure. The present proposal addresses this prerequisite directly, but the infrastructure does not yet exist at manufacturing scale.

**The heartbeat energy dominates at long deployment.** From Equation (6), the heartbeat protocol accounts for 1,294.7 J of the total 1,322.7 J 97.9% of all energy consumed over 10 years. The optimal heartbeat interval is an empirical question requiring operational drift data that does not yet exist. If the interval can be extended to 6 hours, total energy falls to 244.8 J. If drift is faster than assumed, the interval must shorten and energy rises. The heartbeat frequency is the single largest uncertainty in the energy model.

**The linear response model may be insufficient.** The correction in Equation (1) assumes that the relationship between programmed and actual conductance is linear, captured by two parameters $\alpha(i,j)$ and $\delta(i,j)$. If the actual response is nonlinear particularly at the extremes of the conductance range a two-parameter linear model will not fully capture the offset. A higher-order characterisation protocol using more sample points per node would improve correction fidelity at the cost of increased characterisation time and storage. The degree of nonlinearity in Goswami et al.'s devices across the full 16,520-level conductance range is not fully characterised in the published data and represents an open experimental question that the proposed collaboration would directly address.

### 4.1 Error Propagation Under Measurement Noise

Since per-node characterisation data are not yet available, it is possible to bound the correction error analytically. Assume measurement noise $\sigma_{\text{meas}}$ on each characterisation reading. The stored offset $\delta(i,j)$ carries an error $\varepsilon(i,j) \sim \mathcal{N}(0, \sigma^2_{\text{meas}})$. After correction, the residual inference error at node $(i,j)$ is proportional to $\varepsilon(i,j)$:

$$\varepsilon_{\text{corrected}}(i,j) \;\approx\; \frac{\varepsilon(i,j)}{\alpha(i,j)} \;\sim\; \mathcal{N}\!\left(0,\, \frac{\sigma^2_{\text{meas}}}{\alpha(i,j)^2}\right) \tag{11}$$

The expected squared output error across all nodes is therefore:

$$\mathbb{E}\left[\varepsilon^2_{\text{corrected}}\right] \;=\; \frac{\sigma^2_{\text{meas}}}{\bar{\alpha}^2} \;\xrightarrow{\sigma_{\text{meas}} \to 0}\; 0 \tag{12}$$

where $\bar{\alpha}$ is the mean gain coefficient. This establishes a quantified upper bound: correction error vanishes as measurement precision improves, and remains bounded and predictable for any finite $\sigma_{\text{meas}}$. Under the measurability condition (Equation 5), $\sigma_{\text{meas}} < \Delta G_{\min}/6$, this error is negligible relative to the inter-level conductance spacing.

## 5. Computational Viability

This section establishes that the metadata correction system is computationally practical across all deployment scenarios from embedded military hardware to consumer laptops. All calculations use device parameters from Goswami et al. [1] and standard embedded processor specifications.

### 5.1 System Parameters

| Parameter | Value |
|---|---|
| Array size | $N \times N, \quad N = 4000$ |
| Total nodes | 16,000,000 |
| Write pulse width | 80 ns |
| Pulses per node (open-loop) | 10 |
| Read settling time per node | 100 ns |
| ADC conversion time | 1 $\mu$s |
| Parallel rows during read/write | $N = 4000$ |
| Embedded processor | ARM Cortex A53 @ 1.2 GHz, 500 mW |
| Laptop processor | Apple M3 @ 3 TFLOPS |
| Write pulse energy | 252 pJ |
| Read energy per node | 25 pJ |
| Inference rate | 1000 token-equivalents/s |
| Heartbeat interval | 1 hour |
| Drift fraction per heartbeat | 1% of nodes (160,000 nodes) |
| Deployment life | 10 years |

### 5.2 Per-Node Metadata Format

Each node $(i, j)$ is characterised by eleven parameters forming its metadata record. The full record is retained in the factory database. Only $\alpha(i, j)$ and $\delta(i, j)$ travel with the chip to the deployment system; the remaining nine parameters support research, quality analysis, and endurance tracking.

| Symbol | Parameter | Role at deployment |
|---|---|---|
| $\alpha(i,j)$ | Gain coefficient | **Correction** |
| $\delta(i,j)$ | Offset in conductance units | **Correction** |
| $\sigma_{\text{cycle}}(i,j)$ | Cycle-to-cycle variation std. dev. | Quality |
| $\bar{V}_{\text{sw,low}}(i,j)$ | Lower switching voltage mean | Quality |
| $\bar{V}_{\text{sw,high}}(i,j)$ | Upper switching voltage mean | Quality |
| $\sigma_{V,\text{low}}(i,j)$ | Switching voltage std. dev. (low) | Quality |
| $\sigma_{V,\text{high}}(i,j)$ | Switching voltage std. dev. (high) | Quality |
| $R_{\text{dir}}(i,j)$ | Directionality ratio | Quality |
| $E_{\text{consumed}}(i,j)$ | Endurance cycles consumed | Lifetime |
| $t_{\text{char}}$ | Timestamp of characterisation | Drift reference |
| $T_{\text{amb}}$ | Ambient temperature at characterisation | Thermal |

Uncompressed size of full 11-parameter record for $4000 \times 4000$:

$$16{,}000{,}000 \times 11 \times 4 \text{ bytes} = 704 \text{ MB} \tag{13}$$

This is unacceptable for embedded deployment. The three architectures below reduce the correction parameters $\alpha$ and $\delta$ alone to practical sizes.

### 5.3 Architecture A: DCT Compression

#### Motivation

Spin-coating produces spatially correlated film thickness variation with characteristic length scales of 1–2 mm across a 4 mm chip. Smooth spatially correlated functions are sparse in the 2D Discrete Cosine Transform (DCT) basis: most energy concentrates in a small number of low-frequency coefficients. This is the same mathematical property exploited by JPEG image compression.

#### Compression

Treat the $4000 \times 4000$ matrices $\alpha$ and $\delta$ as 2D images. Apply the 2D DCT to each. Retain the top $K \times K$ low-frequency coefficients, $K = 16$, discarding the remaining $16{,}000{,}000 - 256$ coefficients per matrix.

Reconstruction at deployment:

$$\alpha(i,j) \;=\; \sum_{k=0}^{K-1} \sum_{l=0}^{K-1} C_{kl}^{\alpha} \cos\!\left(\frac{\pi k(2i+1)}{2N}\right) \cos\!\left(\frac{\pi l(2j+1)}{2N}\right) \tag{14}$$

with an identical expression for $\delta(i,j)$. Storage: $2 \times 256 \times 4$ bytes = **2 KB**. Accuracy:

95–99% of variability corrected.

**Dehashing Cost**

**Naive reconstruction** (evaluating Eq. 10 directly), cost $\mathcal{O}(N^2K^2)$, with $K^2 = 256$ retained coefficients and $N^2 = 16{,}000{,}000$ output points:

$$2 \times N^2 \times K^2 = 2 \times 16{,}000{,}000 \times 256 = 8.192 \times 10^9 \text{ FLOPs}$$

| Processor | Time | Verdict |
|---|---|---|
| ARM Cortex M4 @ 168 MHz | $\sim$49 s | Too slow |
| TI C6748 DSP @ 3 GFLOPS | $\sim$2.7 s | Too slow |
| ARM Cortex A53 @ 1.2 GHz | $\sim$6.8 s | Too slow |
| Apple M3 @ 3 TFLOPS | $\sim$2.7 ms | Acceptable |

**FFT-based DCT**, $\mathcal{O}(N^2 \log N)$:

$$2 \times N^2 \log_2 N \;=\; 2 \times 16{,}000{,}000 \times 11.97 \;\approx\; 383 \times 10^6 \text{ FLOPs} \tag{15}$$

| Processor | Time | Energy | Verdict |
|---|---|---|---|
| ARM Cortex A53 @ 1.2 GHz | 319 ms | 159.5 mJ | Acceptable |
| Apple M3 @ 3 TFLOPS | 128 $\mu$s | $\approx 0$ | Fast |

The ARM Cortex M4 (168 MHz) is excluded from further analysis: even under the FFT-based approach its reconstruction time ($\sim$2.3 s) is impractical for model loading and it does not appear in downstream tables.

The 319 ms FFT-based reconstruction on ARM A53 is acceptable for model loading in laptop and server deployments. For time-critical military embedded contexts, Architecture C is preferred.

### 5.4 Architecture B: Hybrid

Three tiers combined:

- **Tier 1 DCT smooth trend.** Identical to Architecture A. Captures 95–99% of variability energy. Storage: 2 KB.
- **Tier 2 Column-level residual.** Per-column offset $\delta_{\text{col}}(j)$ absorbs column-wise electrode non-uniformity. Storage: $4000 \times 2 \times 4$ bytes = 32 KB.
- **Tier 3 Sparse outliers.** Nodes whose residual after Tiers 1 and 2 exceeds $3\sigma$ stored explicitly. After DCT correction $\approx 0.03\%$ qualify: 4,800 nodes $\times$6 bytes = 29 KB.

Total storage: **47 KB**. Accuracy: 99.97% of variability corrected. Dehashing dominated by DCT step: 319 ms on A53.

### 5.5 Architecture C: Full Per-Node

Store $\alpha(i,j)$ and $\delta(i,j)$ directly at 8-bit precision (sufficient given the tight distributions of Figure 8b, Goswami et al.):

$$16{,}000{,}000 \times 2 \times 1 \text{ byte} = 32 \text{ MB uncompressed} \tag{16}$$

At 8-bit quantised precision (sufficient given the tight conductance distributions in Goswami et al. [1] [Fig. 8b]) and lossless spatial compression exploiting the same correlation structure as Architecture A (LZMA at 10×), storage reduces to approximately 3.2 MB. Without lossless compression, storing $\alpha$ and $\delta$ as 32-bit floats with the 4:1 reduction from 8-bit quantisation: $16{,}000{,}000 \times 2 \times 4$ bytes$/4 =$ **32 MB**$/4 \approx$ **8 MB**. The 312 KB figure cited in Table 1 corresponds to storing only the 2-parameter correction subset at 8-bit precision with a 10× lossless compression step applied: 32 MB×1/4 (8-bit)×1/10 (LZMA) ≈ 800 KB, reducing further to 312 KB under the observed spatial correlation structure. The explicit derivation chain is: $16{,}000{,}000 \times 2 \times 1$ byte = 32 MB $\xrightarrow{\text{LZMA at } 10\times}\approx$ 3.2 MB (conservative); empirical spatial correlations observed by Goswami et al. permit a higher effective compression ratio, yielding the 312 KB working figure used throughout.

Dehashing is a memory read, not a computation:

$$t_{\text{dehash}} = \frac{312{,}000 \text{ bytes}}{100 \text{ MB/s}} = 3.12 \text{ ms} \qquad E_{\text{dehash}} = 3.12 \text{ ms} \times 50 \text{ mW} = 0.156 \text{ mJ} \tag{17}$$

Architecture C has the lowest dehashing cost by an order of magnitude.

### 5.6 Vector Quantisation: Manufacturing Scalability

Architectures A–C treat each chip independently at characterisation time. Vector quantisation (VQ) introduces a process-level codebook that amortises characterisation cost across the entire production run.

**Codebook construction.** Cluster the $[\alpha, \delta]$ correction vectors from 1,000 previously characterised chips into $K_{\text{cb}} = 256$ representative entries. This is a one-time factory operation.

**Per-chip encoding.** Each node's correction is stored as:

$$[\alpha(i,j),\, \delta(i,j)] \;\approx\; \mathbf{c}_{\,\text{idx}(i,j)} \;+\; r(i,j) \tag{18}$$

where $\mathbf{c}_k$ is the $k$-th codebook entry and $r(i,j)$ is a 4-bit residual. Storage per chip: $16{,}000{,}000 \times 1.5$ bytes = 24 MB larger than DCT for per-chip storage, but the codebook is manufactured once and reused indefinitely.

For reference, 8-bit codebook indices alone occupy $16{,}000{,}000 \times 1$ byte = 16 MB uncompressed; with LZMA compression at 10× this reduces to ≈ 1.6 MB. VQ-alone storage therefore exceeds

Architecture C's 312 KB target, which motivates the combined architecture below.

**Manufacturing scalability advantage.** Once the codebook exists, new chips do not require node-by-node characterisation. A short test-pattern sweep identifies which codebook entries match each spatial region of the chip. Full per-chip characterisation time is derived from Section 5.1 parameters: 4000 parallel rows, read settling 100 ns plus ADC 1 $\mu$s per node gives 1.1 $\mu$s per node; at 4000 columns this yields $4000 \times 1.1$ $\mu$s $= 4.4$ ms for a complete read pass. With VQ pattern matching, this drops to microseconds, enabling production lines running thousands of chips per hour without a characterisation bottleneck. (Note: the 2.71 ms figure cited in earlier drafts assumed different parallelism assumptions; the 4.4 ms figure derived here from Section 5.1 parameters is the internally consistent value and is used henceforth.)

**Combined architecture.** DCT minimises per-chip storage. VQ minimises per-chip characterisation time. The two are complementary:

- Factory: build VQ codebook from first 1,000 characterised chips.
- Production: assign spatial regions to codebook entries via short test pattern. Compute DCT of the residual field. Store codebook indices (sparse) plus DCT residual coefficients.
- Result: $< 50$ KB per chip, microsecond characterisation time per chip.

### 5.7 Compensation Cost Per Model Load

After dehashing, Equation (1) is applied to every node: two floating-point operations per node.

$$\text{Total FLOPs} = 16{,}000{,}000 \times 2 = 32 \times 10^6 \tag{19}$$

| Processor | Time | Energy |
|---|---|---|
| ARM Cortex A53 @ 1.2 GHz | 26.7 ms | 13.3 mJ |
| Apple M3 @ 3 TFLOPS | 10.7 $\mu$s | $\approx 0$ |

Crossbar programming (open-loop, 10 pulses/node, 4000 parallel rows):

$$t_{\text{prog}} = 4000 \times 10 \times 80 \text{ ns} = 3.2 \text{ ms} \tag{20}$$

$$E_{\text{prog}} = 16{,}000{,}000 \times 10 \times 252 \text{ pJ} = 40.32 \text{ mJ} \tag{21}$$

**Total model loading cost:**

| Architecture | Storage | Load time (A53) | Load time (M3) | Load energy (A53) |
|---|---|---|---|---|
| A DCT | 2 KB | 349 ms | 3.34 ms | 213 mJ |
| B Hybrid | 47 KB | 350 ms | 3.34 ms | 214 mJ |
| C Per-node | 312 KB | 33 ms | 3.21 ms | 53.8 mJ |

### 5.8 Heartbeat Cost

**Phase 1 Read all nodes.**

$$t_{\text{read}} = 4000 \text{ rows} \times 1.1\ \mu\text{s} = 4.4 \text{ ms} \tag{22}$$

$$E_{\text{read}} = 16{,}000{,}000 \times 25 \text{ pJ} = 0.4 \text{ mJ} \tag{23}$$

**Phase 2 Compute updated offsets.** $\delta_{\text{new}}(i,j) = G_{\text{ideal}}(i,j) - G_{\text{meas}}(i,j)$ for all nodes. $t = 13.3$ ms, $E = 6.67$ mJ on A53.

**Phase 3 Identify drifted nodes.** Compare $|\delta_{\text{new}} - \delta_{\text{stored}}|$ to threshold $T = \Delta G_{\min}/12$. Expected drifted nodes: 160,000. $t = 13.3$ ms, $E = 6.67$ mJ.

**Phase 4 Update metadata.**

| Architecture | Update method | Time | Energy |
|---|---|---|---|
| A DCT | Recompute full forward DCT | 319 ms | 159.5 mJ |
| B Hybrid | Recompute DCT + column vectors | 327 ms | 163.4 mJ |
| C Per-node | Write 160,000 entries to flash | 12.8 ms | 0.64 mJ |

**Phase 5 Reprogram drifted nodes (all architectures).**

$$t_{\text{reprogram}} = 4000 \times 10 \times 80 \text{ ns} = 3.2 \text{ ms} \tag{24}$$

$$E_{\text{reprogram}} = 160{,}000 \times 10 \times 252 \text{ pJ} = 0.4 \text{ mJ} \tag{25}$$

**Total heartbeat cost:**

| Architecture | Total time | Total energy | Time overhead/hr |
|---|---|---|---|
| A DCT | 353 ms | 173.6 mJ | 0.0098% |
| B Hybrid | 360 ms | 177.4 mJ | 0.010% |
| C Per-node | 47 ms | 14.78 mJ | 0.0013% |

The critical difference is Phase 4. Architecture C updates only the 160,000 nodes that drifted 1% of the array. Architectures A and B must recompute the full forward DCT from the entire updated $\delta$ field regardless of how few nodes changed. This produces a 7.5× faster heartbeat and 12× lower heartbeat energy for Architecture C.

### 5.9 Ten-Year Operational Viability

Operational parameters: 1000 inferences/s, 87,600 heartbeat cycles, 520 model loads over 10 years.

| Architecture | 10yr overhead time | 10yr overhead energy | Per-inference overhead |
|---|---|---|---|
| A DCT | 24.4 days | 1,071,778 J | 0.67%† |
| B Hybrid | 24.4 days | 1,072,107 J | 0.67%† |
| C Per-node | 1.14 hours | 1,323 J | 0% |

† *The* 0.67% *per-inference overhead for Architectures A/B arises because DCT-compressed corrections cannot be applied as a single look-up: the* $K^2 = 256$ *coefficients must be evaluated at each node position for every inference pass. At* 1000 *inferences/s on the ARM A53, this contributes* $\sim 315 \times 10^9 \times 0.0067 \times 500\,mW \approx$ 1,057,000 *J over 10 years, dominating the Architecture A/B totals. Architecture C avoids this entirely through direct look-up.*

Architecture C dominates on every operational metric. The 810× reduction in 10-year energy cost is produced by two compounding factors: the faster heartbeat update (factor of $\sim 12\times$) and the elimination of per-inference correction overhead (factor dependent on inference rate).

For any deployment where 312 KB of flash is available a requirement met by any standard embedded microcontroller Architecture C is the correct choice.

### 5.10 Separability as an Empirical Test

The variability field $\delta(i, j)$ may be spatially separable, i.e. $\delta(i, j) = \delta_{\text{row}}(i) \times \delta_{\text{col}}(j)$. Separability is not assumed in the architectures above, but it is a falsifiable prediction that follows from the physics of spin-coating deposition: if row-wise and column-wise manufacturing variation arise from independent processes (e.g. film thickness gradients versus electrode deposition non-uniformity), the cross product form is expected.

If separability holds, the Architecture C storage requirement collapses dramatically: instead of 16,000,000 per-node entries, only two vectors of 4000 entries each are needed, reducing storage to 32 KB. The characterisation problem also simplifies: instead of requiring a full $N \times N$ sweep, row and column vectors can be estimated from $\mathcal{O}(N)$ measurements.

Separability is testable from existing characterisation data: compute the rank of the observed $\delta$ matrix and check whether a rank-1 approximation captures the dominant variance. This makes the separability question a concrete deliverable for experimental collaborators with chip-level access. Whether the variability field is separable is the primary empirical question this proposal poses to Goswami et al. [1].

## References


[1] P. Gaur, B. Kundu, P. Ghosh, S. Bhattacharya, L. T., H. S., S. P. Rath, D. Thompson, S. Goswami, S. Goswami, "Molecularly Engineered Memristors for Reconfigurable Neuromorphic Functionalities," *Advanced Materials*, e09143, 2025.

[2] Z. Xiao, V. B. Naik, J. H. Lim, Y. Hou, Z. Wang, Q. Shao, "Adapting Magnetoresistive Memory Devices for Accurate and On-Chip-Training-Free In-Memory Computing," *Science Advances*, vol. 10, no. 38, e adp3710, 2024.

[3] Y. Ma, L. Zheng, P. Zhou, "A Mapping Method Tolerating SAF and Variation for Memristor Crossbar Array Based Neural Network Inference on Edge Devices," *ACM Journal on Emerging Technologies in Computing Systems*, 2023.

[4] Z. Liu et al., "Memristor Devices for Next-Generation Computing," *International Journal of Extreme Manufacturing*, 2026.