Adaptive Domain Models: Bayesian Evolution, Warm Rotation, and Principled Training for Geometric and Neuromorphic AI

Houston Haynes
SpeakEZ Technologies, Asheville, NC
hhaynes2@alumni.unca.edu

March 2026

Abstract

Prevailing AI training infrastructure assumes reverse-mode automatic differentiation over IEEE-754 arithmetic. The memory overhead of training relative to inference, optimizer complexity, and structural degradation of geometric properties through training are consequences of this arithmetic substrate. This paper develops an alternative training architecture grounded in three prior results: the Dimensional Type System and Deterministic Memory Management framework [6], which establishes stack-eligible gradient allocation and exact quire accumulation as design-time verifiable properties; the Program Hypergraph [8], which establishes grade preservation through geometric algebra computations as a type-level invariant; and the b-posit 2026 standard [10], which makes posit arithmetic tractable across hardware targets conventionally considered inference-only. Their composition enables depth-independent training memory bounded to approximately twice the inference footprint, grade-preserving weight updates, and exact gradient accumulation, applicable uniformly to loss-function-optimized and spike-timing-dependent neuromorphic models. We introduce *Bayesian distillation*, a mechanism by which the latent prior structure of a general-purpose model is extracted through the ADM training regime, resolving the data-scarcity bootstrapping problem for domain-specific training. For deployment, we introduce *warm rotation*, an operational pattern in which an updated model transitions into an active inference pathway without service interruption, with structural correctness formalized through PHG certificates and signed version records. The result is a class of domain-specific AI systems that are smaller and more precise than general-purpose models, continuously adaptive, verifiably correct with respect to the physical structure of their domains, and initializable from existing models.

1. Introduction

1.1 The Substrate Assumption

IEEE-754 floating-point arithmetic was standardized in 1985. The practice of training deep neural networks at scale emerged roughly in 2012. In the intervening 27 years, floating-point hardware became extraordinarily fast, and practitioners of the emerging deep learning discipline inherited the arithmetic as a fixed property of the landscape. Nobody chose IEEE-754 for neural network training on its merits for that application. It was what computers did with real-valued computation, and the entire ecosystem of training algorithms, optimizers, regularization techniques, and architectural conventions accumulated on top of it.

The result is a field that is *inured* to the error properties of its arithmetic substrate in a specific and measurable sense. The Adam optimizer [13] smooths gradient noise through exponential moving averages. Gradient clipping prevents divergence driven by accumulated rounding error in deep networks. Learning rate warmup stabilizes early training against gradient variance that has a precision component. Batch normalization and layer normalization compensate partly for the accumulation of rounding error across layers. Mixed-precision training (bfloat16 forward, float32 accumulation) is an engineering accommodation to the fact that float16 accumulation produces unacceptable gradient noise while float16 computation is faster. Each of these techniques is genuinely effective and theoretically motivated. Each also partially functions as precision compensation for arithmetic whose error properties were never designed with gradient-based learning in mind.

The consequence that is directly relevant to this paper concerns geometric structure. IEEE-754 arithmetic makes the structural zeros of a Clifford algebra Cayley table numerically non-zero through accumulated rounding. A bivector weight initialized at grade 2 accumulates small grade-0, grade-1, grade-3, and grade-4 components through gradient updates. The self-reinforcing consequence is that the network learns to compensate for its own grade corruption, occupying a training basin that incorporates the arithmetic's error profile as a structural feature rather than exploiting the algebraic properties the architecture was designed to provide. This is why Clifford algebra neural networks, despite their demonstrated theoretical advantages [18, 22], have not achieved widespread adoption: the training process destroys the properties that make them useful, and no existing framework provides a substrate that preserves those properties through training.

This paper develops that substrate.

1.2 The Three Enabling Results

Three prior results compose into the training architecture this paper describes.

The first is the Dimensional Type System and Deterministic Memory Management framework established in [6]. That work established forward-mode automatic differentiation [3] as a training modality with a specific and verifiable coeffect signature: no activation tape, $O(1)$ auxiliary memory per layer, and the inner product in the directional derivative computable exactly via the quire accumulator. The quire provides exact dot product accumulation, rounding once after full accumulation rather than once per multiply-add operation. The combination eliminates two distinct sources of gradient error: the heap allocation and tape management overhead of reverse-mode, and the accumulated rounding in gradient inner products. Both are consequences of IEEE-754 arithmetic, not of learning.

The second is the Program Hypergraph established in [8]. That work demonstrated that grade in Clifford algebra is a type-level invariant under the DTS framework's dimensional group structure, that grade inference determines the non-zero entries of the geometric product Cayley table at design time, before any arithmetic occurs, and that PHG saturation semantics correctly formalize multi-way geometric constraints that binary edge graphs cannot represent without information loss. The critical extension for training: the dual-number representation used in forward-mode autodiff augments each primal multivector with a tangent component of the same grade. PHG grade inference applies identically to tangent computations. A grade-2 bivector weight has a grade-2 gradient by the chain rule closure of grade-preserving operations, enforced as a design-time constraint enforced by the type system. Grade corruption through training is a type violation, surfacing in the Lattice language server as the code is written. Numerical drift in IEEE-754 would cause the same corruption to accumulate silently; the type system surfaces it immediately.

The third is the b-posit 2026 standard [10]. The bounded regime field (limited to ≤ 6 bits) reduces posit decoder hardware cost to 79% less power, 71% less area, and 60% less latency relative to standard posit decoders, achieving cost competitive with IEEE-754 float32 decoders on NPU-class hardware. This is the threshold at which posit arithmetic becomes tractable on edge and embedded hardware targets. Below this threshold, posit arithmetic required data center hardware to justify its precision advantages. At this threshold, posit arithmetic with quire accumulation is a realistic substrate for AI training on devices that also run inference.

A consequence of the bounded regime design that receives less attention than the precision profile is the hardware multiplexer unification it enables. With $r_S = 6$, the regime field is always between 2 and 6 bits in length, five possibilities selectable by a single MUX. Combined with the sign bit, the non-significand portion of a posit word has a maximum width of $1 + r_S + e_S$ bits. As Gustafson establishes [9], this structure allows a single hardware implementation to serve 16-bit, 32-bit, and 64-bit posit operations, since the regime and exponent fields scale uniformly. IEEE-754 formats cannot share hardware across precisions because the exponent field width and bias differ between float16, float32, and float64 in ways that require separate decode logic. The b-posit bounded regime eliminates this cost: one decoder, one MUX, three precisions.

1.3 Contributions

This paper makes five claims.

1. **The Program Hypergraph (PHG) constitutes a natural substrate for Bayesian posterior inference over geometric model parameters.** The Dimensional Type System (DTS) dimensional annotations and PHG grade constraints express a structural prior over model parameters before any training data is observed: the prior is the type system, encoding domain knowledge as type-level constraints rather than as regularization penalties. Forward-mode autodiff with quire accumulation provides the posterior update mechanism. Distribution shift, measured as KL divergence between the model's current predictive distribution and the empirical distribution of recent operational observations, provides a principled, domain-calibrated trigger criterion for incorporating new evidence into an updated model. The correctness conditions for this update, and the infrastructure for deploying it without interrupting active inference, are developed in Section 4.

2. **Through this mechanism, Clifford algebra neural networks achieve grade preservation through training as a theorem, with exact equivariance and stable sparsity as direct corollaries.** Grade preservation follows from PHG grade inference and the dual-number coeffect signature of forward-mode autodiff: a grade-k weight has a grade-k gradient by the chain rule closure of grade-preserving operations, enforced as a design-time constraint. Exact rotor equivariance follows because rotor normalization is verified at design time through the SMT-LIB2 proof infrastructure. Cayley table

sparsity is stable across the full training history as a corollary of grade invariance. Together these properties make Clifford algebra networks, with their demonstrated equivariance advantages for physical simulation and geometric learning, a structurally superior substrate for domain-specific AI on the b-posit arithmetic foundation.

3. **General-purpose language models carry latent Bayesian prior structure that the ADM training regime can extract and formalize, providing a tractable bootstrapping path for domain prior initialization.** Large models trained on scientific and technical corpora absorb statistical regularities about domain relationships, physical constraints, and probabilistic reasoning patterns. This latent structure constitutes an unstructured prior: accessible but not formally constrained. The ADM training regime acts as a distillation mechanism: the DTS dimensional annotations and PHG grade constraints filter the extracted prior, retaining what is dimensionally and geometrically coherent with the target domain and discarding what is not. The output is a domain model whose prior provenance is traceable to the general model but whose structural properties are formally certified by the type system. We term this *Bayesian distillation*. It resolves the data-scarcity bootstrapping problem that domain-specific training otherwise faces, and it positions general-purpose models as prior sources within the ADM architecture rather than as alternatives to it. Recent empirical work demonstrating that latent Bayesian structure is accessible in general LLMs [24] establishes the precondition on which this mechanism depends.

4. **Spike-timing-dependent plasticity and forward-mode autodiff share a common local learning signature, and the Fidelity framework's type system provides a unified verification and deployment infrastructure for both.** The design-time verification, the Deterministic Memory Management (DMM) stack-eligible allocation discipline, the quire accumulation semantics, and the versioning infrastructure established for gradient-descent models apply to STDP-trained spiking networks without modification. A spiking network trained via STDP on a neuromorphic processor and a Clifford algebra network trained via forward-mode on a spatial dataflow accelerator are instances of the same adaptive domain model architecture, sharing the same formal correctness properties while differing in hardware target and temporal representation.

5. **The actor model provides a principled compositional architecture between the rigid homogeneity of Mixture of Experts and the structural informality of current agentic AI frameworks.** The PHG establishes at design time the inter-domain constraints and dimensional properties that each actor carries into runtime as structural invariants of its compiled form. Clef's actor model, in which Prospero supervises domain computation units (Olivier actors), composes these verified actors into a heterogeneous system where domain boundaries are enforced by the structure of the actors themselves. BAREWire carries dimensional annotations across inter-actor message boundaries, making violations detectable at the message fabric level.

1.4 Relation to Prior Work

This paper is the third in a sequence. The DTS/DMM paper [6] established the foundational type system, memory management discipline, and forward-mode autodiff analysis. The

PHG paper [8] extended this to multi-way geometric constraints, grade-typed Clifford algebra computation, and spatial dataflow architectures. This paper takes the PHG's grade preservation properties and the DTS/DMM's forward-mode coeffect analysis as established results and develops their implications for a training architecture that operates continuously, at depth-independent training memory cost, on domain-specific geometric and neuromorphic models.

Throughout this paper, the term *design time* refers to the period when an engineer is writing source code, before any build step is invoked and before any execution occurs. In the Fidelity framework, the Composer compiler runs continuously as a language server process through Lattice, elaborating the program's type constraints incrementally as source is edited and surfacing diagnostics in the development environment as they are discovered. This is distinct from what practitioners in ML frameworks typically mean by "compile time," which commonly refers either to a discrete build invocation or to JIT kernel compilation that occurs at execution onset. A grade violation, a dimensional inconsistency in a loss term, or a boundary mismatch in an inter-actor message: each of these surfaces as a diagnostic at design time in the Fidelity framework, before any build is initiated and before any training run is attempted. The significance for model development is that the class of errors described in this paper are not discovered by running experiments; they are excluded by the structure of the program as it is written.

2. The IEEE-754 Inurement Argument

2.1 Precision Loss Overhead as Training Overhead

The memory overhead of neural network training relative to inference is commonly quoted as a factor of three to ten, depending on model size, optimizer choice, and mixed-precision configuration. This figure is calibrated on reverse-mode automatic differentiation with IEEE-754 arithmetic, and it is treated in the field as a near-universal constant. It is not. It is a consequence of specific arithmetic choices, and identifying those choices precisely is necessary before proposing an alternative.

Reverse-mode autodiff requires storing every intermediate activation from the forward pass for use in the backward pass. For a network with L layers and batch size B, this is $O(L \cdot B)$ storage that serves no purpose during inference and is discarded after each training step. The activation tape is a feature of computing the gradient efficiently via the chain rule over a stored computation graph, one that forward-mode autodiff with random projection eliminates: it computes an unbiased gradient estimate in a single forward pass with $O(1)$ auxiliary memory per layer [3]. The tape cost is an artifact of the reverse-mode algorithm, separable from the learning objective itself.

The optimizer state cost is a second component. Adam [13] maintains first and second moment estimates of the gradient for each parameter, doubling the parameter storage. The moment estimates serve two purposes: they reduce gradient variance (a learning benefit) and they smooth gradient noise (a precision compensation benefit). With exact quire accumulation, the precision compensation component of moment estimation is unnecessary. The remaining benefit, variance reduction, is achievable with simpler online statistics at lower storage cost.

Gradient clipping is the most direct form of precision compensation: it truncates gradient magnitudes that have grown large through accumulated rounding, preventing weight updates from driving parameters into degenerate regions. A gradient that is large because the loss landscape is steep in a particular direction is informative and should not be clipped. A gradient that is large because accumulated rounding errors have compounded across layers is noise. In IEEE-754 these are indistinguishable without additional diagnostic machinery. With quire exact accumulation, the gradient magnitude reflects the loss landscape rather than the arithmetic, and the diagnostic need for clipping as a safety mechanism is substantially reduced.

2.2 Grade Corruption as a Training Failure Mode

The inurement argument has a specific geometric form in the context of Clifford algebra neural networks. A weight matrix W of declared grade k is updated by the gradient $\nabla_W \mathcal{L}$. In IEEE-754, the gradient computation involves accumulation of products along the backward pass, and the accumulated value has rounding error deposited at grades other than k. After n training steps, the weight W is no longer a pure grade-k element:

$$W^{(n)} = W_k^{(0)} + \sum_{j \neq k} \epsilon_j^{(n)} e_j$$

where $W_k^{(0)}$ is the grade-k initialization, e_j are basis elements at grade $j \neq k$, and $\epsilon_j^{(n)}$ are accumulated rounding contributions that grow with n. The contamination is systematic, not random: the same arithmetic path produces the same rounding bias on each forward pass, so $\epsilon_j^{(n)}$ is correlated across steps rather than averaging out.

The forward pass through a contaminated weight produces contaminated activations. The contaminated activations propagate to the next layer and beyond. The backward pass through contaminated activations produces contaminated gradients. The contaminated gradients update the weights in directions that partially compensate for the activation contamination. The network converges to a basin where contamination and compensation are in equilibrium, which is a different basin from the one the clean algebraic structure would have produced.

The practical consequence was identified in Section 3.4 of [8]: runtime sparsity detection cannot distinguish a structurally zero Cayley entry (zero by the algebra's rules, regardless of inputs) from an entry that is small due to accumulated rounding. The Flash Clifford implementation [21] addresses this by manually hardcoding the non-zero entries, but the manually hardcoded sparsity applies only to the initialized weight structure, not to the trained weight structure. After training in IEEE-754, the trained weights have non-zero contributions at entries that were structurally zero at initialization, and the manual hardcoding is no longer valid. The sparsity advantage, the primary computational motivation for Clifford algebra networks, cannot survive training in IEEE-754.

2.3 What Changes with the Principled Substrate

With PHG grade inference, forward-mode autodiff, and quire accumulation, the grade corruption mechanism is eliminated at the source rather than managed after the fact.

Grade inference at the type level establishes which Cayley entries are structurally zero before any arithmetic occurs. These entries have no arithmetic path; they are absent from the compiled computation, not merely small in value. The quire provides exact accumulation for the arithmetic paths that do exist, so the non-zero entries are computed without accumulated rounding. The dual-number extension of the forward pass carries grade-typed tangent components through the same type-level constraints as the primal: the gradient of a grade-k weight is certified grade-k by the compiler with the same formal guarantee as any other type annotation in the system.

The training process, under this substrate, is grade-preserving by construction. The weight $W_k^{(n)}$ after n training steps is certified grade-k for all n; the type system makes any other grade an error at design time, visible in the Lattice language server as the code is written. The trained model's sparsity is identical to the initialized model's sparsity. The computational advantage is stable across the full training history and through any number of updates under our warm rotation model.

3. The Bayesian Training Model

3.1 Posterior Update as the Principled Alternative

Gradient descent in the standard formulation is a point estimate method: the model maintains a single set of weights and moves them in the direction of the negative gradient. The method works because the loss landscape, empirically, tends to have many good minima and the optimizer finds one of them. What the method does not provide is a calibrated measure of uncertainty: the trained model produces outputs but cannot reliably indicate when those outputs are in a region of its input space that is poorly covered by training data.

That this is a consequential and empirically confirmed gap is supported by recent empirical work. Van Steenkiste and Linzen [24] demonstrate that off-the-shelf large language models fail to exhibit genuine Bayesian updating behavior: their performance on a sequential recommendation task, where the optimal strategy requires updating a belief distribution over user preferences across five rounds, plateaus after a single interaction regardless of subsequent evidence. The optimal symbolic Bayesian model in their study achieves 81% accuracy; unmodified LLMs cluster significantly below this and show limited improvement over multiple interactions. The authors address this through supervised fine-tuning on demonstrations from the Bayesian model, a procedure they term "Bayesian teaching," and find that the resulting models generalize improved Bayesian behavior to unseen domains. The empirical conclusion is that Bayesian reasoning structure does not emerge spontaneously from scale or general pre-training; it must be introduced deliberately. The question of mechanism, how that structure is introduced and what properties it carries, is where the present paper diverges from that line of work, as discussed in Section 8.

A principled Bayesian training model maintains a posterior distribution over the model's parameters rather than a point estimate. Given a prior $p(\theta)$ encoding structural knowledge about the domain (which, in the Fidelity framework, is expressed in part through the DTS dimensional annotations and PHG grade constraints) and observed data $\mathcal{D}$, the posterior is:

$$p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta)\, p(\theta)$$

The prior $p(\theta)$ is not uniform. The DTS dimensional type annotations encode the physical structure of the domain: a weight connecting a velocity field layer to a pressure gradient layer carries a dimensional prior $\langle \mathrm{m\,s^{-1}} \rangle \to \langle \mathrm{Pa\,m^{-1}} \rangle$ that constrains the posterior regardless of the data. The PHG grade constraints encode the geometric structure: a grade-2 bivector weight has a prior that concentrates probability mass at grade 2, enforced as a type constraint rather than as a regularization penalty.

Forward-mode autodiff with the quire provides a natural mechanism for posterior approximation. The directional derivative $\langle \nabla_\theta \mathcal{L}, v \rangle$ for a random unit vector v is an unbiased scalar estimate of the gradient projected onto v. Under a Gaussian posterior approximation, repeated directional derivative estimates with independent random projections accumulate into a low-rank estimate of the Hessian's eigenvectors, which together with the gradient magnitude provide the curvature information needed for a natural gradient update. The quire's exact accumulation means each directional derivative estimate is computed without the numerical noise that contaminates Hessian estimates in IEEE-754, making the posterior approximation more accurate for a given number of observations.

3.2 Distribution Shift as the Adaptation Trigger

A model deployed in an operational context observes data drawn from the operational distribution, which may differ from the training distribution. Under a Bayesian framing, this is simply new evidence: the posterior over model parameters should be updated as operational data accumulates. The question is not whether to update but when the accumulated evidence is sufficient to justify a verified model update.

The appropriate trigger criterion is the KL divergence between the model's current predictive distribution and the empirical distribution of recent operational observations:

$$D_{\mathrm{KL}}\big(p_{\mathrm{empirical}} \parallel p_{\mathrm{model}}\big) > \epsilon_{\mathrm{domain}}$$

where $\epsilon_{\mathrm{domain}}$ is a domain-calibrated threshold. For a structural health monitoring model, the threshold is calibrated in terms of sensor confidence intervals and the dimensional ranges established in the DTS annotations. For a fluid dynamics model, it is calibrated in terms of regime transition indicators. The dimensional type annotations make the threshold semantically grounded in the domain rather than architecturally generic.

This trigger criterion is distinct from a degradation signal in both direction and interpretation. A degradation signal is backward-looking: the model performed worse on recent data, indicating a problem to be corrected. A distribution shift trigger is forward-looking: new information is available that would make the model more accurate, and the warm rotation mechanism defined in Section 4 provides the operational path to incorporate it. The model's state is defined by its most recent posterior update: a weight configuration current as of time t_0, scheduled for revision at time t_1 when sufficient new evidence has accumulated.

Figure 1 shows the full adaptive cycle from prior specification through operational evidence accumulation to verified model update and re-entry into inference.

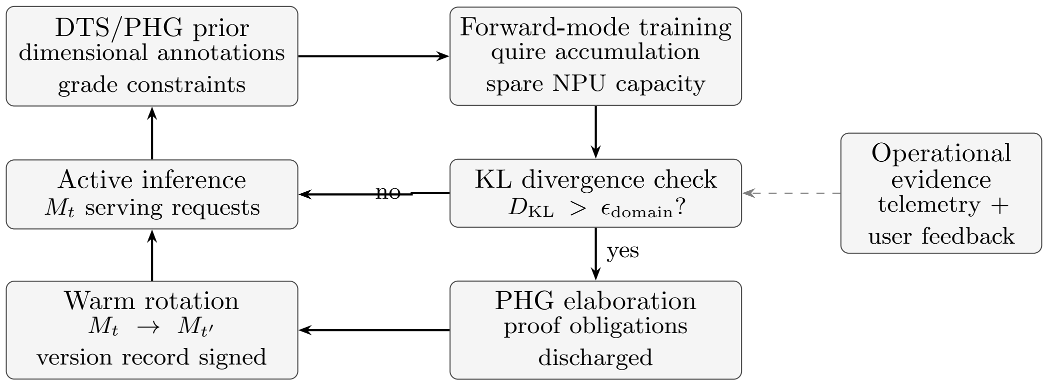

Figure 1: The adaptive domain model cycle. The DTS and PHG annotations constitute the structural prior. Forward-mode training in spare compute capacity produces a candidate posterior. Operational evidence (telemetry and user feedback) drives the KL divergence check. When the threshold is crossed, PHG elaboration discharges proof obligations before warm rotation. Below-threshold observations accumulate without triggering a rotation.

3.3 Consumer-Supplied Evidence and Operational Provenance

The model's operational context provides two classes of evidence that are unavailable at initial training time: sensor measurements, telemetry, and derived observations from the deployment environment (which we term *operational evidence*), and user-supplied preference judgments on model outputs (which constitute a specific form of likelihood signal in the Bayesian model).

User preference judgments, when expressed in the context of a specific model output over a specific input, are observations about the model's predictive accuracy in a particular region of the input space. They update the posterior over the model's parameters in that region. This is not RLHF in the conventional sense, which constructs a separate reward model from preference data and uses it to fine-tune via a policy gradient. It is a direct Bayesian likelihood update: the user's judgment that output y was incorrect for input x is an observation that $p(y \mid x, \theta)$ should be low, which constrains the posterior over θ. The update is local to the relevant region of parameter space, dimensional type-compatible with the domain, and incorporated into the same warm rotation pipeline as operational sensor evidence.

The versioning infrastructure records the provenance of all evidence contributing to a model update. A version record carries: the KL divergence value at trigger time, the sources of the operational evidence (sensor types, time windows), any user preference observations with their input contexts, the PHG structural certificate of the updated weight configuration, and the post-quantum signature anchoring the record to the keys of the operating organization. A consumer of the model system can verify not only what structural properties the current model has but why each version transition occurred and what evidence drove it.

3.4 Bayesian Distillation from General-Purpose Models

The Bayesian training model described in this section treats the structural prior as something to be constructed: dimensional type annotations and PHG grade constraints encode domain knowledge before any training data is observed. In practice, constructing a well-founded prior for a specialized physical domain from first principles requires either sufficient domain-specific training data or a hand-crafted symbolic model of the domain's structure. Both prerequisites may be scarce in the early stages of deploying an adaptive domain model.

A tractable alternative is available. Large general-purpose language models, trained on scientific literature, engineering documentation, and technical corpora at scale, absorb statistical regularities about physical relationships, dimensional constraints, causal dependencies, and probabilistic reasoning patterns. This accumulated structure constitutes a latent prior: not formally specified, not dimensionally annotated, and not geometrically coherent in the sense the DTS and PHG require, but present and accessible.

Recent empirical work [24] establishes the key precondition: latent Bayesian structure in general-purpose LLMs is accessible through targeted fine-tuning and generalizes across domains. This result, developed independently of the ADM framework, confirms that the prior knowledge absorbed during large-scale pretraining is not merely a surface statistical regularity but a structural property that can be elicited and directed.

We propose *Bayesian distillation* as the mechanism that makes this accessible structure useful within the ADM architecture. The process operates in three stages. In the extraction stage, a general-purpose model is queried over a target domain's problem space: its outputs, uncertainty characterizations, and internal activation patterns are collected as an unstructured empirical prior distribution over domain parameters. In the formalization stage, this distribution is passed through the ADM training regime: the DTS dimensional annotations act as a filter, retaining probability mass consistent with the domain's dimensional constraints and attenuating mass that violates them; the PHG grade constraints perform the same function for geometric structure. In the certification stage, the resulting weight configuration is elaborated by Composer, PHG proof obligations are discharged, and the distilled prior is registered as a version record with provenance traceable to the source model.

The output is a domain model whose prior was seeded from the breadth of a general model's training but whose structural properties are formally certified by the type system. The distillation regime does not copy the general model's outputs; it extracts the probabilistic structure implicit in those outputs and imposes the domain's type-level constraints as a formalizing filter. What survives is a prior that is both informed by large-scale pretraining and verifiably consistent with the domain's physical structure.

This mechanism resolves the data-scarcity bootstrapping problem in a principled way and repositions general-purpose models within the ADM architecture: not as alternatives to domain-specific models but as prior sources that the distillation regime formalizes. The general model contributes breadth; the type system contributes precision; the combination produces a domain model that is both informed and certifiable.

4. The Warm Rotation Architecture

The term *warm rotation* is introduced in this paper as a novel operational pattern. ML practitioners will recognize the cold-start penalty: a model that has been unloaded from memory requires full reinitialization before it can serve inference requests, a process that incurs latency and compute cost. Warm rotation describes a distinct and more demanding property: the managed exchange of active model weights while inference continues, such that no request observes a service interruption and the incoming configuration has been structurally verified before it enters the pathway. The analogy to a cold start is deliberate; what we are specifying here is the architectural conditions under which the transition can be made without one. Those conditions derive from properties of the training algorithm and the type system, not from any specific silicon architecture. The warm rotation mechanism is hardware-agnostic in the same sense that forward-mode autodiff is hardware-agnostic: the relevant properties hold wherever inference is feasible, across CPU, GPU, NPU, CGRA, and neuromorphic targets alike.

The architecture described in this section represents active design work. The formal structure is established here as a specification: the correctness conditions, the memory feasibility argument, and the relationship to the PHG elaboration and versioning infrastructure are developed with sufficient precision to guide implementation. The implementation itself, across the full range of hardware targets and deployment topologies described, is in progress. Where the text uses present-tense language to describe system behavior, it should be read as describing design intent.

4.1 Formal Definition

Let M_t denote the model running in the active inference pathway at time t. M_t consists of a weight configuration θ_t, a PHG structural certificate $\mathcal{C}(\theta_t)$ produced by the Composer compiler's elaboration pass, and a version record $\mathcal{V}_t$ signed by the organization's post-quantum credential.

Definition 4.1 (Warm Rotation)**.** A *warm rotation* is a transition $M_t \to M_{t'}$ where:

1. $M_{t'}$ was trained using available compute capacity, whether concurrent with active inference, during periods of reduced ambient utilization, or during any interval in which the hardware budget is not fully committed to inference;
2. $\mathcal{C}(\theta_{t'})$ is present and valid: the Composer compiler has elaborated the updated weight configuration and discharged all SMT-LIB2 proof obligations for dimensional consistency, grade correctness, and representation adequacy;
3. $\mathcal{V}_{t'}$ records the training provenance, the distribution shift trigger, and the structural certificate, signed by the organization's post-quantum key;
4. the transition is atomic with respect to inference requests: no request observes a partial state during the model exchange.

Condition (1) establishes that warm rotation does not require taking the inference pathway offline, nor does it require a dedicated training window. The qualifying condition is simply that the training computation for $M_{t'}$ completes before the rotation is initiated: whether that computation ran concurrently alongside inference, during off-hours when utilization was low, or opportunistically across intermittent idle intervals is an operational scheduling decision that the architecture accommodates without modification. Crucially, the training computation need not occur on the same hardware as active inference. A side-load deployment, whether a container provisioned at the network edge, a home server receiving a bootstrap instruction over a secure channel, or any compute node reachable within the operating organization's trust boundary, can receive the training task and return a candidate weight configuration for PHG elaboration and rotation. The memory parity of forward-mode training relative to inference is the hardware feasibility condition that makes co-location viable when it is convenient: a device running M_t in inference at k TOPS has $(\text{Total} - k)$ TOPS available for concurrent training, and the same device at rest has its full compute budget available for accelerated training. The architecture does not require this co-location; it merely permits it, which is the distinguishing property. In either case the memory footprint of training is bounded to approximately twice the inference footprint, making the training computation a well-behaved workload that can be suspended, resumed, and migrated across available compute resources without architectural special-casing.

Condition (2) is the correctness condition that distinguishes warm rotation from unverified model replacement. The PHG structural certificate is a formal proof that the updated weight configuration satisfies the structural invariants the architecture requires, grounded in the elaboration pass's discharged SMT-LIB2 obligations. A weight configuration that fails PHG elaboration cannot be rotated into the active pathway.

Conditions (3) and (4) are the trust and consistency conditions. The version record anchors the transition to its evidence chain. The atomicity requirement is implementable through the Olivier model's message-passing semantics: the active model actor processes all in-flight requests before acknowledging the rotation, and new requests are buffered and replayed against $M_{t'}$ after the transition.

4.2 Memory Feasibility Across Hardware Targets

The memory parity argument developed in Section 4.1 holds across hardware targets because it derives from the absence of the activation tape, which is a property of the training algorithm rather than a property of any specific execution substrate. Whether the inference pathway runs on a general-purpose CPU, a GPU, an NPU tile array, a CGRA, or a neuromorphic processor, the forward-mode training pass requires approximately twice the inference memory footprint, independent of the silicon architecture. This is an algorithm-level property that hardware adaptation can accelerate but cannot change in character. The framework's feasibility argument therefore applies to any target for which inference is viable, without modification for each architecture class.

Figure 2 illustrates the memory and compute allocation during a warm rotation event on a representative 50 TOPS inference-class accelerator.

The critical observation is that the spare capacity (30 TOPS) is sufficient for training because the training memory footprint is bounded to a fixed constant multiple of the inference memory footprint, independent of model depth. In reverse-mode training, the candidate model $M_{t'}$ would require the activation tape in addition to the forward pass memory: total training

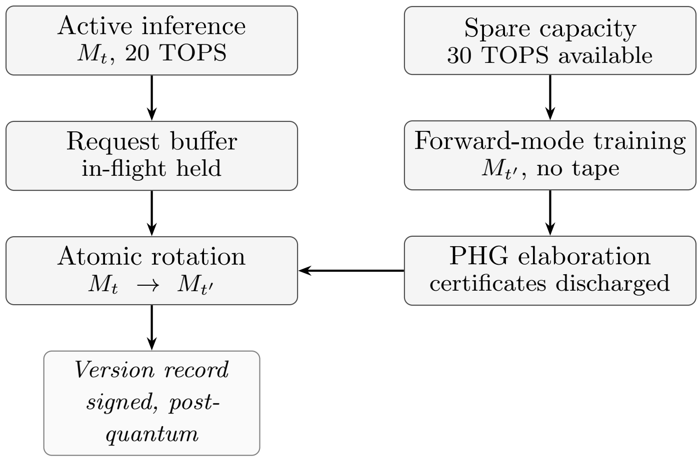

Figure 2: Warm rotation on a representative 50 TOPS inference-class accelerator. Active inference occupies 20 TOPS. Forward-mode training of the candidate model requires approximately 2× the inference memory footprint (primal plus tangent component), consuming roughly 20 TOPS of the spare 30 TOPS. PHG elaboration discharges structural certificates before the atomic rotation. In-flight requests are buffered and completed against the new model.

memory is $(1 + L)$ times inference memory for a network with L layers. With forward-mode training, total training memory is approximately 2× inference memory (primal plus tangent component), regardless of depth. For any device where inference is feasible, forward-mode training is feasible within approximately double the inference memory budget.

The b-posit arithmetic has a direct consequence for the spare capacity calculation. The near-unity precision advantage of posit arithmetic means that the effective precision of the forward-mode gradient estimate, for activations concentrated near unity (which is the typical case for normalized network activations), is higher than an IEEE-754 float32 estimate at the same bit width. The gradient estimate requires fewer random projections to achieve a target variance level, which reduces the effective compute cost of the training pass. On a b-posit-capable accelerator, the 30 TOPS available for training is more productive per TOPS than the equivalent IEEE-754 compute would be.

The choice of posit parameters for ML workloads warrants specific attention. Gustafson addresses this directly in Chapter 13 of *Every Bit Counts: Posit Computing* [9], where the parameterization for application-specific number systems is laid out in full. The conclusion relevant here is that $e_S = 5$, appropriate for general scientific computing with its wide dynamic range, is unnecessary for ML inference and training: the dynamic range required is approximately $[10^{-14}, 10^1]$, far narrower than the $[10^{-18}, 10^{18}]$ range of general HPC workloads. The NUS work on 8-bit bounded posit demonstrated that the optimal ML parameterization uses asymmetric e_S and r_S values across the posit ring: values with magnitude less than 1 (the lower half of the ring diagram) use different parameters from values with magnitude greater than or equal to 1 (the upper half), biasing the precision distribution toward the near-unity region where normalized activations concentrate. The exponent bias is set so that the central high-precision region is at 2^{-2} or 2^{-3} rather than at unity, placing the maximum relative accuracy exactly where the activation distribution peaks during training. The same NUS experiments established a lower bound of 5 bits for classification accuracy: reducing to 4 bits produces a sharp degradation in network accuracy, independent of the parameterization. This 5-bit floor is a property of the information content required for gradient signal propagation, not of the posit format specifically, and it informs the minimum word width for the adaptive domain model training substrate described in this paper.

A standard objection to inference hardware deployment concerns inference-time quantization: large models must be post-training quantized to INT8, INT4, or lower precision to fit available memory, and this quantization is inherently lossy because the model was trained at higher precision than the deployment target supports. Approaches such as BitNet [25] address this pressure by training in ternary weights from the outset, designing the model for its hardware constraint rather than compressing it afterward. The approach is principled and the results are competitive precisely because the model learns within its precision constraint rather than having that constraint imposed on weights that were never optimized for it.

ADM models face this deployment pressure differently, because the sources of inference-time memory overhead are structurally different from the start. The quantization pressure on large general models arises from density: a dense parameter matrix requires high precision per element to represent the full range of distinctions the model must make across all domains in its training corpus. An ADM model's computational structure is sparse before any precision question arises. The PHG grade inference eliminates structurally zero Cayley table entries from the compiled computation entirely; they are absent from the inference artifact, not compressed within it. The 85 to 95 percent sparsity for 3D PGA computations cited in Section 5.1 is not a pruning ratio achieved by removing small weights; it is the fraction of computations that were never instantiated because the type system proved them unnecessary. The b-posit parameterization then concentrates arithmetic precision in the near-unity region where the domain's activation distribution actually lives, as established by the NUS experiments above, rather than distributing uniform precision across a range most activations never occupy.

The combined effect is that a well-parameterized ADM model achieves an inference memory footprint competitive with aggressively quantized large models in the same domain, without lossy precision reduction, because the structural properties of the domain were encoded into the computation graph at design time rather than learned from data and then compressed out. BitNet and the ADM approach address the same deployment pressure from orthogonal starting points: BitNet accepts the dense-model paradigm and reduces precision radically through principled training; ADM reduces the structural density of the computation before the precision question arises. The two approaches are composable: a 1-bit or ternary ADM model would compound the memory reduction from structural sparsity with the precision reduction of BitNet-style training. Whether this combination preserves accuracy across physical domains is an open empirical question, but the compositional path is architecturally clean.

4.3 Cluster-Scale Warm Rotation

The warm rotation argument extends naturally to deployments where multiple domain actors are distributed across a cluster of compute nodes rather than co-located on a single device. The memory parity property that makes warm rotation feasible on a single device applies equally in the distributed case: each actor's training computation requires approximately twice its inference memory footprint, and that computation can be scheduled across available cluster nodes with the same flexibility described in Section 4.1.

Distributed gradient accumulation across cluster workers is an accumulation of directional derivative estimates, precisely the reduction the quire makes exact. Current practice uses bfloat16 reduction and relies on statistical averaging across workers to manage rounding error. Quire-backed reduction is exact regardless of worker count, rounded once after full accumulation. For training runs spanning large accelerator pools and many steps, this eliminates a systematic source of gradient bias that current frameworks treat as irreducible. The versioning infrastructure applies uniformly: each distributed training run produces a candidate weight configuration that must pass PHG elaboration before a warm rotation is initiated, and the version record anchors the distributed provenance of the training evidence regardless of how many nodes contributed to the gradient accumulation.

5. Clifford Neural Networks Under the Principled Substrate

5.1 Grade Preservation Through Training

Section 3.2 of [8] established that grade inference derives the non-zero Cayley table entries for geometric products at design time, and that PHG saturation verifies the grade constraints of each geometric operation before any arithmetic occurs. Section 2 of this paper established that IEEE-754 gradient accumulation corrupts the grade structure of trained weights in a systematic and self-reinforcing way.

The composition of these two results yields the following:

Proposition 5.1 (Grade Invariance of Forward-Mode Training)**.** Let W be a weight parameter with declared grade k in a PHG-typed Clifford neural network. Let $\nabla_W \mathcal{L}$ be the gradient of the loss with respect to W, computed via forward-mode autodiff with dual-number augmentation and quire accumulation. Then $\nabla_W \mathcal{L}$ has grade k as a a design-time certificate from the type system, and the updated weight $W' = W - \eta \nabla_W \mathcal{L}$ has grade k for all learning rates η and all inputs.

Proof sketch. The dual-number extension of a grade-k primal is a grade-k tangent. PHG grade inference applies identically to the tangent computation as to the primal. The directional derivative $\langle \nabla_W \mathcal{L}, v \rangle$ is a scalar (grade-0) result, computed as the quire accumulation of products between the grade-k weight tangent components and the random projection v. The gradient $\nabla_W \mathcal{L}$ is the reconstruction of the weight-space vector from the scalar projection estimates; under the PHG type system, this reconstruction is constrained to the grade-k subspace. Any component at grade $j \neq k$ is a type violation caught at design time. $\square$

Corollary 5.1 (Sparsity Stability)**.** The non-zero Cayley table entries determined by PHG grade inference at compile time are the same after n training steps as at initialization, for all n.

This corollary is the formal statement of the practical claim: a Clifford neural network trained under the Fidelity framework retains its sparsity advantage across the full training history. The 85% to 95% sparsity reported for 2D and 3D PGA computations in [21] is a structural invariant of the trained model, preserved by the grade type system at every training step.

5.2 Equivariance as a Structural Guarantee

The sandwich product $X \mapsto RX\tilde{R}$, where R is a rotor satisfying $R\tilde{R} = 1$, is exactly equivariant under the Lie group of the algebra's metric by construction. For PGA ($\mathbb{R}^{3,0,1}$) this is SE(3); for STA ($\mathbb{R}^{1,3,0}$) it is the Lorentz group.

In IEEE-754, a "rotor" weight initialized to satisfy $R\tilde{R} = 1$ accumulates odd-grade contamination through gradient updates and ceases to be an exact rotor after training. The sandwich product then provides approximate equivariance, which is a statistical regularity the network has preserved under the training dynamics rather than a structural guarantee. For distribution shifts that expose regions of the input space not well-covered by training data, the approximate equivariance may fail.

Under the PHG grade type system and forward-mode training, a rotor weight is a type-constrained even-grade element. The unit norm constraint $R\tilde{R} = 1$ is a dimensional constraint in the DTS sense: it is an invariant over the algebra's inner product that the SMT-LIB2 verification infrastructure can express and discharge at elaboration time. Rotor normalization is verified at design time, and the equivariance of the trained model is exact by structural guarantee for all inputs.

5.3 Mixed-Algebra and Physics-Structured Architectures

Grade preservation enables network architectures whose grade requirements IEEE-754 arithmetic cannot maintain through training. Two cases are directly relevant.

Mixed-algebra networks. A network whose early layers operate in PGA for spatial geometry and whose later layers operate in CGA for sphere and circle detection can be represented in the PHG with algebra transition hyperedges at the boundary. The PHG verifies that the representation conversion from a grade-3 PGA point to its CGA embedding preserves the geometric identity, verifiable at design time. This was not expressible in previous frameworks because no previous framework had grade as a first-class type-level property at the compiler level; the conversion would be a floating-point matrix multiply with no algebraic provenance.

Physics-structured networks. In a physics-informed network where individual layers correspond to physical operators (gradient, divergence, curl, Hodge dual), the grade rule of each operator is a type constraint. The gradient of a scalar pressure field $\langle \mathrm{Pa} \rangle$ is a grade-1 vector field $\langle \mathrm{Pa\,m^{-1}} \rangle$. A layer that accidentally applies a divergence where a gradient is intended is a type error, caught at design time. The trained model's physical operator structure is verifiable in the same elaboration pass that verifies its geometric grade structure.

6. Spiking Neural Networks and the Temporal Hyperedge

6.1 Temporal Sparsity as a PHG Dimension

The spatial sparsity in Clifford algebra computation (grade-determined Cayley zeros) and the temporal sparsity in spiking neural networks (silence between spikes) are instances of the same structural property in the PHG, expressed in different dimension axes.

Clifford sparsity is grade-dimensional: the PHG grade annotation determines which computation nodes are absent before arithmetic occurs.

Neuromorphic temporal sparsity is time-dimensional: the PHG temporal annotation on spike events determines which computation nodes are active at which times. At any moment, only a small fraction of neurons in a spiking network fire; the remainder are silent and consume negligible power. This temporal sparsity is the source of the three-order-of-magnitude energy efficiency advantage of Loihi-2 relative to GPU for matched workloads, for the same reason that Clifford spatial sparsity is the source of the $20\times$ arithmetic reduction: doing nothing, exactly, is different from computing a small value that is then discarded.

In the DTS, time is a dimension axis with annotation $\langle\mathrm{ms}\rangle$. The membrane time constant τ_m of a LIF neuron carries this annotation. The synaptic weight carries the derived annotation $\langle\mathrm{mV\,ms}^{-1}\rangle$. A spiking network whose temporal dynamics are dimensionally inconsistent across layers, where the inter-spike intervals expected by one layer do not match the firing rates produced by the previous layer, is a design-time error under DTS dimensional inference. No current neuromorphic programming framework provides this property.

6.2 Coincidence Detection as a Hyperedge

The Leaky Integrate-and-Fire neuron model is:

$$\tau_m \frac{dV_m}{dt} = -V_m + \sum_i w_i \sum_k \delta(t - t_i^k)$$

where $\tau_m\langle\mathrm{ms}\rangle$ is the membrane time constant, $w_i\langle\mathrm{mV\,ms}^{-1}\rangle$ are synaptic weights, and t_i^k are presynaptic spike arrival times. A neuron fires when V_m reaches threshold $V_\theta\langle\mathrm{mV}\rangle$.

Coincidence detection, the fundamental computational primitive, occurs when k presynaptic inputs spike within temporal window τ. This is a joint property of the full k-tuple of arriving spikes. The PHG represents it as a k-to-1 hyperedge:

$$f = \big(\{s_1, \ldots, s_k\},\ \text{fire},\ \lambda_\tau\big)$$

where λ_τ carries the constraint $\max_i t_i - \min_i t_i \le \tau\langle\mathrm{ms}\rangle$. PHG saturation fires the target node only when all k source spike nodes are present and the temporal constraint is satisfied. Decomposing this into $\binom{k}{2}$ binary edges loses the joint temporal constraint: pairwise co-occurrence does not entail k-way co-occurrence within the window.

6.3 STDP and the Local Learning Coeffect Signature

Spike Timing Dependent Plasticity updates the weight w_{ij} between presynaptic neuron i and postsynaptic neuron j as:

$$\Delta w_{ij} = \begin{cases} A_+ \exp(-\Delta t/\tau_+) & \Delta t > 0 \\ -A_- \exp(\Delta t/\tau_-) & \Delta t < 0 \end{cases}$$

where $\Delta t = t_{\text{post}} - t_{\text{pre}}$ and all time constants carry annotation $\langle \text{ms} \rangle$.

The coeffect signature of STDP is identical in structure to forward-mode autodiff: local parameter updates depending only on information available at the synapse, $O(1)$ auxiliary state per parameter (one trace variable per spike direction), no global error signal, and no activation tape. Loihi-2 implements STDP in hardware via its on-chip learning engine, which maintains trace variables without off-chip memory access. The DMM coeffect discipline verifies that STDP trace variables are stack-eligible with the same machinery that verifies forward-mode tangent components.

Proposition 6.1 (Unified Local Learning Signature)**.** Forward-mode autodiff over loss-function models and STDP over spiking neural networks share a common coeffect signature: (a) no global error signal or activation tape; (b) $O(1)$ auxiliary state per parameter; (c) update computations that are stack-eligible under DMM coeffect inference; (d) inner products or accumulations that are exact under quire semantics where the hardware capability coeffect is available.

The proposition establishes a structural classification: the two learning rules are computationally analogous in their coeffect signatures. Both achieve gradient-free (in the tape sense) local learning with bounded auxiliary state, and both have their accumulation operations made exact by the quire. The warm rotation and versioning infrastructure, as specified in Section 4, is designed to apply uniformly to both modalities.

6.4 Hybrid Geometric-Neuromorphic Networks

Grade-typed Clifford representations and spike-coded neuromorphic representations are complementary in a specific way. Clifford algebra excels at encoding continuous geometric structure (spatial relationships, physical fields, manifold topology) with exact algebraic properties. Spiking representations excel at encoding temporal patterns, event sequences, and sparse coincidence structure with extreme energy efficiency.

A hybrid network with Clifford algebra layers for geometric feature extraction and spiking layers for temporal pattern recognition is a single PHG with both grade annotations (on Clifford nodes) and temporal annotations (on spike nodes). The interface between the two regimes is a representation conversion hyperedge: a Clifford layer's continuous-valued grade-1 output is encoded as a spike rate or timing code for the neuromorphic layer's input. The PHG verifies that this conversion is dimensionally consistent: the spike rate annotation carries a dimensional annotation derived from the Clifford layer's output dimension and the encoding function's dimensional structure.

The PHG's per-target reachability bitvector handles the hardware partitioning: Clifford nodes are XDNA-reachable (route to MLIR-AIE lowering), spike nodes are Loihi-2-reachable (route to NxCore API emission), and the conversion nodes are CPU-reachable or implement a DMA transfer with dimensional type preservation via BAREWire. This is the compilation model for a heterogeneous AI Engine that uses each substrate for what it does best, from a single shared intermediate representation. Figure 3 illustrates the PHG structure and hardware partitioning for this hybrid architecture.

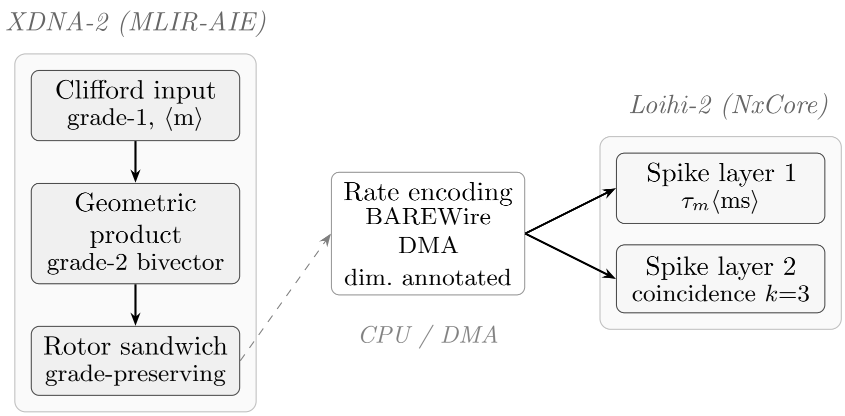

Figure 3: PHG structure of a hybrid geometric-neuromorphic network. Clifford algebra nodes (left, XDNA-2 target) carry grade and dimensional annotations. A rate-encoding conversion hyperedge (centre, CPU or DMA via BAREWire) bridges to spiking layers (right, Loihi-2 target) carrying temporal and coincidence annotations. The per-target reachability bitvector in the PHG routes each subgraph to its appropriate lowering path from a single intermediate representation.

7. The Versioning and Trust Infrastructure

7.1 PHG Certificate Differencing

PHG certificate differencing is a design-stage capability: the formal machinery for producing and comparing structural certificates exists within the Composer elaboration pipeline, but the versioning layer that records, stores, and surfaces certificate diffs across model transitions is under active development. What follows describes the intended behavior of this infrastructure and the properties it is designed to provide.

Each model version is intended to carry a PHG structural certificate: a set of elaborated annotations covering dimensional consistency, grade correctness, topological consistency, co-location feasibility, and representation adequacy. A version transition $M_t \to M_{t'}$ would produce a certificate diff: the set of annotations that changed between versions.

Under our warm rotation model, a transition driven by new training data in a specific physical domain produces a certificate diff that identifies exactly which subgraph of the PHG changed and in what respect. If a force-balance expert added a thermal coupling term, the diff shows that a new inter-domain hyperedge was added between the force and thermal subgraphs, with the specific dimensional constraint it encodes. This is a human-interpretable description of a functional change, derived from the model's structural properties: a proof-level record of what structural commitments changed, grounded in the elaborated annotations rather than in the weight tensors themselves, which carry no interpretable information about semantic intent.

7.2 Post-Quantum Version Signing

The formalism applied to version signing is intended to match the stakes of the deployment environment, not to impose a uniform requirement across all uses of the architecture. At the minimal end, a warm rotation record may carry nothing more than a timestamp, a hash of the updated weight configuration, and the PHG certificate diff, sufficient for a developer tracking their own local model updates with no adversarial threat model. At the other end, a deployment in a contested environment, whether a critical infrastructure system, an air-gapped clinical device, or a security-sensitive organizational context, requires a record whose authenticity can be verified by any party with access to the organization's public key, whose integrity is durable against cryptographic capabilities not yet available today, and whose provenance chain is auditable without trusting any single intermediary. The architecture is designed to support both ends of this range and the continuous gradient between them: the version record is a structured object whose signing policy is a deployment configuration, scaled to the security posture the operating environment requires.

For deployments operating under a Zero Trust security model, version records are intended to be signed by the operating organization's post-quantum key via the QuantumCredential infrastructure. The signature anchors the version record to the key at the time of the warm rotation, making the record tamper-evident against adversaries with access to current and anticipated future cryptographic capabilities.

The intended trust chain from training data through model structure through inference output is as follows: operational evidence is collected with timestamped provenance; the Bayesian update is triggered when the KL divergence threshold is crossed; the forward-mode training run produces a candidate weight configuration; Composer elaborates the configuration and discharges PHG proof obligations; the version record is constructed from the training provenance, the PHG certificate, and the certificate diff; the record is signed according to the deployment's configured signing policy; the warm rotation is executed. When fully realized, every step in this chain would be recorded and independently verifiable at the level of assurance the deployment requires. A consumer of the model system would be able to verify the structural properties of the running model, the evidence that drove the last version transition, and the integrity of the version record, without running the model against adversarial test cases.

Contested deployment environments require a trust model grounded in demonstrable provenance. The version record described here provides exactly that: an auditable chain, anchored to the organization's key policy, that answers the questions such environments demand. What is the model structurally capable of? What evidence drove the last transition? When did it occur, and under whose authority? Benchmark performance against a curated test set cannot answer these questions. A cryptographically signed, PHG-diffed version record can.

7.3 Scope Bounds and the World Model Framing

A domain-specific model under the Fidelity framework is bounded in scope by its dimensional type annotations and PHG grade constraints. These bounds are not limitations to be overcome; they are the properties that make the model trustworthy. A structural health monitoring model cannot learn to perform medical diagnosis because the dimensional constraints of its domain do not encompass medical measurement types. A fluid dynamics model cannot produce outputs in units of currency because the DTS would catch the dimensional inconsistency at the point where such an output was declared.

This scope boundedness addresses the objection that domain-specific models are merely "small models" insufficient for real-world complexity. The relevant comparison is not parameter count but the correctness of the model's outputs within its problem domain. For any problem with physical structure, an adaptive domain model whose dimensional annotations and grade constraints are proven consistent is strictly more reliable than a large general model for that problem, not because it is larger but because its structural properties are verified rather than approximated. General-purpose large models distribute their capacity across all domains and verify nothing. Adaptive domain models concentrate their capacity and carry verified structural properties throughout their operational lifetime.

Composability restores the breadth that scope boundedness removes. Clef is a concurrent programming language in the ML family whose actor model is central to its design. In this model, Prospero is the supervisor actor responsible for orchestration and arena lifetime management; Olivier actors are the domain computation units that Prospero oversees. A deployed system composes multiple Olivier actors, each running a domain model scoped to its annotated problem space, with inter-actor communication carried through BAREWire with dimensional type annotations preserved across message boundaries. The PHG hyperedge structure encodes the inter-domain constraints, verifiable by the same saturation machinery that verifies intra-domain constraints within each actor. The aggregate capability of such a system is broad by composition, while each component retains the verified properties of its domain: dimensional consistency, grade correctness where applicable, and elaborated structural certificates.

The scope boundedness and closed-system training properties of this architecture have a specific consequence for organizations whose operational data carries sensitivity constraints. An organization whose domain models are trained exclusively on internally curated data, whose training provenance is recorded in cryptographically signed version records, and whose model updates are triggered by distribution shift within a bounded operational context, rather than by general corpus updates from external sources, retains full custodianship of the knowledge embedded in its models. Proprietary operational data, trade-sensitive process knowledge, and curated internal information corpora do not need to leave the organization's trust boundary at any stage of the training, distillation, or rotation pipeline. The Bayesian distillation mechanism allows such an organization to seed its domain priors from general-purpose models without exposing its operational data to those models; the warm rotation mechanism ensures that model updates are verified against the organization's own evidence provenance before entering the active pathway. The parametric integrity of the resulting models, that their structural properties are proven consistent with the domains they were trained on and the data that drove each version transition, is a property the organization can demonstrate to its own auditors, regulators, or counterparties without disclosing the underlying data itself. This is the architecture that makes private AI tractable as an engineering discipline rather than an aspiration bounded by the capabilities of hosted general models.

This is the architecture through which genuinely ambient, genuinely trusted AI becomes an engineering challenge with a clear solution path.

8. Related Work

8.1 Bayesian Teaching and Behavioral Approximation

Van Steenkiste and Linzen [24] establish that Bayesian reasoning behavior in large language models requires deliberate structural introduction, and that large-scale pretraining installs latent Bayesian structure that can be elicited through targeted fine-tuning. Their Bayesian teaching framework constructs a symbolic optimal Bayesian model for a given task domain, generates fine-tuning data from that model's outputs, and trains the LLM to approximate those outputs. The resulting models generalize improved Bayesian behavior across domains, including domains not seen during fine-tuning.

The contribution is empirically significant and the result establishes a precondition on which Section 3.4 of this paper depends: that latent Bayesian structure in general-purpose LLMs is accessible and transferable. Their finding that this structure generalizes across domains is what makes Bayesian distillation tractable as a prior initialization mechanism for the ADM architecture. In this sense, the two bodies of work are complementary rather than parallel: Bayesian teaching demonstrates that the latent prior is real and accessible; Bayesian distillation proposes to extract and formalize it under the structural constraints of a target domain's type system.

The architectural distinction between the two approaches remains important. Bayesian teaching produces behavioral approximation: the fine-tuned model learns to produce outputs that correlate with an optimal Bayesian agent's outputs on the training distribution. No formal prior structure is recoverable from the LLM's weight space after fine-tuning, no posterior update procedure is certifiable, and no versioning record connects the model's current behavior to the evidence that shaped it. The ADM Bayesian distillation mechanism uses the general model's latent structure as a starting point and then imposes the DTS and PHG constraints as a formalizing filter, producing a domain model whose structural properties are proven at design time and whose provenance is cryptographically anchored. The general model contributes the prior's breadth; the type system contributes its precision and certifiability.

The Bayesian assistant that Van Steenkiste and Linzen construct is, in the language of this paper, a domain model: a computational agent whose parameters are a probability distribution over domain features, updated by Bayes' rule as evidence arrives. The structural identification is exact. The Bayesian assistant works because the domain is simple enough to specify completely. The adaptive domain model architecture generalizes this pattern to domains with richer geometric and physical structure, with the prior expressed through the type system, the posterior update implemented through forward-mode autodiff with quire accumulation, and the update trigger formalized as a KL divergence criterion calibrated to the domain's dimensional annotations. For physically structured domains where correctness guarantees matter, the structural approach is the appropriate one; for general-purpose language tasks, behavioral approximation remains viable precisely because the prior structure of natural language does not admit compact formal specification.

8.2 Geometric Algebra Neural Networks

The Clifford Group Equivariant Neural Network (CGENN) work of Ruhe et al. [18] and the Clifford-Steerable CNN work of Zhdanov et al. [22] establish the theoretical and empirical case for geometric algebra as an architectural substrate for equivariant learning on physical simulation tasks. The Flash Clifford implementation [21] addresses the runtime performance gap through fused kernels and warp-aligned memory layout. Section 5 of this paper develops the argument that these approaches are constrained by the IEEE-754 substrate: the grade preservation properties that make Clifford algebra networks equivariant by construction cannot survive training in IEEE-754 arithmetic without manual architectural intervention, and the Cayley table sparsity that provides the computational advantage cannot be reliably exploited when training corrupts the grade structure of weights. The PHG grade type system and forward-mode training substrate resolve this at the architectural level.

8.3 Neuromorphic Programming Frameworks

Intel's Lava framework, PyNN, and NEST provide programming models for spiking neural network hardware at varying levels of abstraction. None provides grade or temporal dimensional annotations as type-level properties, none verifies the consistency of temporal dynamics across network layers, and none integrates with a compilation pipeline that carries semantic information through to hardware configuration. The relationship between STDP's local learning signature and forward-mode autodiff's coeffect structure, established in Section 6.3, is not represented in any existing neuromorphic programming framework. The consequence is that spiking networks and gradient-descent networks are treated as categorically separate architectures requiring separate toolchains, where the present work establishes them as instances of the same adaptive domain model architecture.

8.4 The Bitter Lesson and Its Preconditions

Sutton's bitter lesson [23] crystallizes a pattern documented across machine learning domains for more than fifteen years: general methods that leverage computation and scale consistently outperform hand-crafted, knowledge-engineered approaches given sufficient data. Banko and Brill [2] demonstrated this formally for natural language disambiguation in 2001, showing that the best algorithms at low data underperformed the worst algorithms with orders of magnitude more data. The pattern repeated across image synthesis, speech recognition, game playing, and general reasoning tasks. Halevy, Norvig, and Pereira's survey [19] generalized the finding across domains, and the lesson's recurrence over successive generations of ML practitioners gave it its name.

The ADM framework does not contradict the bitter lesson. It operates in the regime where the lesson's precondition does not hold. The bitter lesson applies with full force to domains where the world model is implicit, vast, and not formally characterizable: natural language, general image understanding, broad reasoning. For these domains, there is no compact formal description of the problem's structure that can substitute for data at scale, and general methods leveraging computation reach higher asymptotes than structured approaches.

For physically structured domains, the precondition is different. The dimensional constraint that force has dimension $\mathrm{kg} \cdot \mathrm{m} \cdot \mathrm{s}^{-2}$ is not a hand-crafted feature to be overcome by data. It is a fact about the physical world that is known with certainty before any training example is observed. Encoding it as a type-level constraint eliminates an entire class of hypotheses from the learning problem, compressing the effective hypothesis space in a way that data cannot improve upon. The DTS and PHG do not compete with scale; they reduce the problem to one where less scale is required by removing from consideration the hypotheses that the formal structure of the domain already rules out. The Bayesian distillation mechanism reflects this directly: the latent prior extracted from a large general model is approximately right about physical domains, and the ADM training regime replaces the approximation with the formally correct prior, raising the effective asymptote without requiring additional data.

8.5 Mixture of Experts, Agentic Frameworks, and the Actor Model

The sparse MoE architecture of Shazeer et al. [20] provides the cluster-scale context for the warm rotation argument in Section 4.3. The observation that forward-mode variance is tractable per active expert when P_{expert} is compact by design, and that distributed gradient accumulation is an exact operation under quire semantics, extends the warm rotation argument from single-device inference hardware to cluster-scale training without architectural modification.

The architectural relationship between the Fidelity actor model and both MoE and current agentic AI frameworks warrants direct characterization, as it locates this work in a space that neither existing paradigm occupies.

Sparse MoE achieves scalable specialization through a learned routing distribution over a structurally uniform expert pool. All experts share the same architecture, parameter scale, and input-output interface. The routing mechanism learns which experts to activate for a given input, but that routing is a statistical approximation: there is no structural guarantee that a given expert will receive inputs consistent with whatever latent specialization it has developed, and no formal mechanism to enforce that each expert's parameter updates respect the dimensional or geometric constraints of its intended domain. Expert collapse is a symptomatic failure of this informality. The Fidelity actor model enforces specialization structurally: each actor's domain is defined by its DTS dimensional annotations and PHG grade constraints at compile time, and inputs that fall outside those constraints are boundary violations detectable at the message fabric level rather than routing failures discovered through degraded accuracy.

Current agentic AI frameworks, including the growing ecosystem of LLM-orchestration and tool-use architectures, achieve compositional flexibility by composing general-purpose models with external tools, memory systems, and other agents through natural language or loosely typed API calls. The correctness of inter-agent interactions is established through prompt engineering, testing, and empirical observation. Domain boundaries between agents are implicit in the natural language of their system prompts rather than enforced by any structural mechanism. This informality is the source of both the frameworks' flexibility and their fragility: a system that relies on a language model's interpretation of "you are a medical diagnosis assistant" for domain enforcement cannot provide the same guarantees as a system where the medical domain's dimensional constraints are enforced by the type system at every message boundary.

The Fidelity actor model occupies the principled position between these two poles. It achieves the heterogeneous specialization that MoE targets, with domain boundaries enforced structurally rather than approximated statistically. It achieves the compositional flexibility that agentic frameworks target, with inter-component constraints made explicit in the PHG at design time rather than left implicit in natural language at runtime. The correctness properties of the composed system are established before deployment, not discovered through operational experience.

9. Future Work

9.1 Bayesian Distillation: Empirical Validation and Scope

The Bayesian distillation mechanism proposed in Section 3.4 requires empirical validation across a range of physical domains and source models. The central open questions are: how much of a general-purpose model's latent prior structure survives the DTS and PHG filter in a given domain; how the quantity and diversity of the extraction queries affect the coherence of the resulting distilled prior; and whether the data-efficiency gains from distillation initialization over cold-start domain training are consistent across domains with differing dimensional and geometric complexity.

Formal metrics for prior coherence before and after distillation are also needed. A distilled prior should be measurably more consistent with the domain's dimensional constraints than the raw extracted distribution, and the improvement should be quantifiable in terms of the DTS constraint satisfaction rate and the PHG saturation convergence properties. Developing these metrics would allow the distillation mechanism to be evaluated objectively and tuned for specific deployment contexts.

A further open question is the relationship between source model scale and distillation quality. The mechanism assumes that larger models with broader training corpora carry richer latent prior structure for physical domains. Whether this assumption holds uniformly, or whether domain-specific pretraining at smaller scale produces a more coherent source for distillation, is an empirical question with significant practical implications for the cost of deploying adaptive domain models in resource-constrained environments.

10. Conclusion

The argument developed in this paper begins with a substrate observation and ends with an architectural claim.

The substrate observation is that the memory overhead of training, the optimizer complexity, and the structural degradation of geometric properties through gradient updates are consequences of IEEE-754 arithmetic accumulated over four decades of engineering practice, not properties of learning itself. The field is inured to these properties because it has always had them and has built effective tools around them. Identifying them as substrate artifacts, separable from the learning problem, is the prerequisite for addressing them.

The architectural claim is that the combination of PHG grade preservation, forward-mode autodiff with quire accumulation, and b-posit arithmetic constitutes a training foundation that eliminates these arithmetic artifacts without sacrificing the statistical properties that make gradient-based learning work. The result is a class of models that train at approximately twice the inference memory footprint, independent of model depth, maintain geometric structural properties through training as type-level invariants, and adapt continuously to their operational context through the warm rotation mechanism introduced in Section 4.

The paper introduces two novel mechanisms developed within this architecture. The first, warm rotation, provides the operational path by which an updated model enters an active inference pathway without service interruption, with structural correctness verified before the transition and the evidence provenance cryptographically anchored in the version record. The second, Bayesian distillation, resolves the prior initialization problem: the latent Bayesian structure of a general-purpose language model, demonstrated to be accessible and transferable by recent empirical work [24], is extracted and formalized through the ADM training regime. General-purpose models contribute the breadth of their training as a prior source; the type system imposes the domain's structural constraints as a formalizing filter. The result is a domain model that is both informed by large-scale pretraining and certifiable with respect to its domain's physical structure.

Both loss-function gradient descent over Clifford algebra neural networks and spike-timing-dependent plasticity over neuromorphic spiking networks are instances of this architecture. The warm rotation model and versioning infrastructure defined in Section 4 is designed to apply uniformly to both. A system of interlocking domain models within a managed actor framework brings together the precision of expert model traversal and the compositional reach of agentic coordination, while asserting verifiable results from each model and integrity of communication at every shared inter-actor boundary. We assert that this combination, structurally enforced specialization with formally verified message boundaries, is the property neither pure MoE nor current agentic frameworks can provide simultaneously.

General-purpose language models participate in this system in two distinct roles, neither of which is terminal. As Bayesian distillation sources, they seed domain prior initialization with the breadth of their pretraining, providing a tractable starting point where domain-specific data is scarce. As active participants in the running system, they serve interim roles during periods when specialized domain actors are still being developed, trained, or refined, contributing general reasoning capability within the dimensional boundaries the actor framework enforces at the message fabric level. As domain expertise matures within the closed system, general-purpose participants yield progressively more of their roles to structurally verified domain actors, but they remain available as a reasoning resource for problem spaces that do not yet warrant or support a dedicated domain model.

The phrase "ambient AI you can trust" describes a structural property: trust established at design time, through elaboration of the program's structural properties, grounded in cryptographically anchored version records rather than in benchmark performance against a curated test set.

Acknowledgments

The author thanks John L. Gustafson for guidance on posit arithmetic, the b-posit 2026 standardization pathway, and for *Every Bit Counts: Posit Computing* [9], whose Chapter 13 provides the definitive parameterization reference for application-specific posit systems. The treatment of representation selection for training arithmetic in Sections 3 and 4 reflects his technical influence directly. The author thanks Barak A. Pearlmutter for his encouragement of this research program and for his foundational work on automatic differentiation, which, together with the forward gradient result in Baydin et al. [3], underpins the training substrate described in this paper. The author also thanks Don Syme, whose F# language and Units of Measure system are the type-theoretic substrate from which DTS draws its inference architecture, and whose co-authorship of the forward gradient paper connected this work to the broader AD research community. The author also thanks Paul Snively for contributions on delimited continuations and the geometric algebra problem framing documented in the companion paper [8] that informed this work.

Software Availability

The Clef language, Composer compiler, and supporting libraries described in this paper are developed under the Fidelity Framework project. Source repositories are available at `https://github.com/FidelityFramework`. The language specification, design rationale, and compiler documentation are published at `https://clef-lang.com`. All components referenced in this paper, including the ADM training substrate, warm rotation infrastructure, and BAREWire interchange protocol, are under active development.

References

[1] AMD/Xilinx. MLIR-AIE: An MLIR-based toolchain for AMD AI engines, 2024. `github.com/Xilinx/mlir-aie`.

[2] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of ACL*, 2001.

[3] A. G. Baydin, B. A. Pearlmutter, D. Syme, F. Wood, and P. Torr. Gradients without backpropagation. *arXiv preprint arXiv:2202.08587*, 2022.

[4] M. Coll. Inet dialect: Declarative rewrite rules for interaction nets. MLIR Open Design Meeting, April 2025.

[5] S. De Keninck, M. Roelfs, L. Dorst, and D. Eelbode. Clean up your mesh! Part 1: Plane and simplex. *arXiv preprint arXiv:2511.08058*, 2025.

[6] H. Haynes. Dimensional type systems and deterministic memory management: Design-time semantic preservation in native compilation. SpeakEZ Technologies, 2026.

[7] H. Haynes. Quantum optionality and the precision problem. Clef Language Framework blog, 2026. `clef-lang.com/blog/quantum-optionality/`.

[8] H. Haynes. The program hypergraph: Multi-way relational structure for geometric algebra, spatial compute, and physics-aware compilation. SpeakEZ Technologies, 2026.

[9] J. L. Gustafson. *Every Bit Counts: Posit Computing*. Chapman and Hall/CRC Computational Science. CRC Press, Boca Raton, FL, 2024. ISBN 978-1-032-73805-5.

[10] S. Jonnalagadda, S. Thotli, and J. L. Gustafson. b-posit: Bounded-regime posit arithmetic for energy-efficient AI hardware. *IEEE Transactions on Emerging Topics in Computing*, 2025. Updated 2026 per working group revision.

[11] B. Kang, H. Desai, L. Jia, and B. Lucia. WAMI: Compilation to WebAssembly through MLIR without losing abstraction. *arXiv preprint arXiv:2506.16048*, 2025.

[12] A. Kennedy. Types for units-of-measure: Theory and practice. In *Central European Functional Programming School*, LNCS 6299. Springer, 2009.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015.

[14] C. Lattner et al. MLIR: Scaling compiler infrastructure for domain specific computation. In *Proceedings of CGO*, 2021.

[15] T. Petricek, D. Orchard, and A. Mycroft. Coeffects: A calculus of context-dependent computation. In *Proceedings of ICFP*, 2014.

[16] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks. *Journal of Computational Physics*, 378:686–707, 2019.

[17] A. Rico et al. AMD XDNA NPU in Ryzen AI processors. *IEEE Micro*, 44(6):73–83, 2024.

[18] D. Ruhe, J. Brandstetter, and P. Forré. Clifford group equivariant neural networks. *arXiv preprint arXiv:2305.11141*, 2023.

[19] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[20] N. Shazeer et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[21] M. Zhdanov. Flash Clifford: Hardware-efficient implementation of Clifford algebra neural networks. `github.com/maxxxzdn/flash-clifford`, 2025.

[22] M. Zhdanov et al. Clifford-steerable convolutional neural networks. In *Proceedings of ICML*, 2024.

[23] R. S. Sutton. The bitter lesson. Incomplete Ideas blog, March 2019. `incompleteideas.net/IncIdeas/BitterLesson.html`.

[24] S. van Steenkiste and T. Linzen. Bayesian teaching enables probabilistic reasoning in large language models. *Nature Communications*, 2026. `doi.org/10.1038/s41467-025-67998-6`.

[25] H. Wang, S. Ma, L. Dong, S. Huang, H. Wang, P. Ma, X. Xia, and F. Wei. BitNet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023.

[26] biVector.net geometric algebra library catalog, 2025. `bivector.net/lib.html`.