

Differentiable Logic Synthesis: Spectral Coefficient Selection via Sinkhorn-Constrained Composition

Gorgi Pavlov, Ph.D.

Lehigh University & Johnson and Johnson

gorgipavlov@gmail.com

February 9, 2026

Abstract

Learning precise Boolean logic via gradient descent remains challenging: neural networks typically converge to “fuzzy” approximations that degrade under quantization. We introduce **Hierarchical Spectral Composition**, a *transparent-by-design* architecture that selects spectral coefficients from an interpretable Boolean Fourier basis and composes them via Sinkhorn-constrained routing with column-sign modulation. Each Fourier coefficient $\hat{f}(S)$ has explicit semantic meaning—the correlation of f with parity of variables S —making learned representations intrinsically explainable. Our approach draws on Manifold-Constrained Hyper-Connections (*mHC*) [1], adapting Birkhoff polytope projection from LLM training stability to logic synthesis, with column-sign modulation enabling Boolean negation.

We validate across five phases: (1–3) For $n = 2, 3, 4$, we achieve 100% accuracy on canonical Boolean operations via gradient descent ($n = 2$), exhaustive enumeration ($n = 3$), and spectral synthesis with MCMC refinement ($n = 4$), all converging to ternary masks $\{-1, 0, +1\}$ enabling single-cycle GPU inference at 10,959 MOps/s. (4) Exact Walsh-Hadamard transforms scale to $n = 28$ (268M coefficients, 1.64B coeffs/sec). (5) **Oracle learning experiments** ($n = 16$) compare five coefficient estimation methods (Monte Carlo, Goldreich-Levin, spectral filtering) on parity, majority, and comparator functions, revealing that *symbolic structure beats black-box learning*: exploiting symmetry constraints boosts majority accuracy from 72% (MC) to 86% (GL+Symmetry), a +38% gain ($p < 0.001$), while Birkhoff projection aids routing but not coefficient denoising. These findings establish that **domain knowledge integration**—encoding known function properties as hard spectral constraints—produces both higher accuracy and greater interpretability than generic learning algorithms, aligning with the vision of explainable neurosymbolic AI.

1 Introduction

The integration of symbolic reasoning with gradient-based learning remains a fundamental challenge in artificial intelligence. Neural networks excel at continuous pattern recognition but struggle with tasks requiring *exact* discrete logic: they approximate Boolean functions with soft decision boundaries that degrade under distributional shift, adversarial perturbation, or—critically for deployment—quantization. Existing neuro-symbolic approaches, such as Neural Arithmetic Logic Units (NALU) [26], Neural Logic Machines [27], Logical Neural Networks [29], or Logic Tensor Networks [28], either require extensive supervision, rely on relaxed continuous operators that resist discretization, or fail to generalize beyond their training distribution.

We propose a different paradigm: **Spectral Selection and Composition**. Rather than learning logic gates from random dense initializations, we ground our architecture in the Fourier analysis

of Boolean functions [3].

Boolean Fourier Analysis. Any function $f : \{-1, +1\}^n \rightarrow \mathbb{R}$ has a unique *Fourier expansion*:

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S(x), \quad \text{where } \chi_S(x) = \prod_{i \in S} x_i \quad (1)$$

and $\hat{f}(S) = \mathbb{E}_x[f(x)\chi_S(x)]$ are the exact Fourier coefficients. When f maps to $\{-1, +1\}$, these coefficients are determined uniquely by the truth table.

Polynomial Threshold Representations. Our architecture learns a different object: a **ternary polynomial threshold function** (PTF):

$$\hat{y}(x) = \text{sign} \left(\sum_{S \subseteq [n]} w_S \chi_S(x) \right), \quad w_S \in \{-1, 0, +1\} \quad (2)$$

The key insight is that for many Boolean functions, there exist *sparse* ternary weight vectors w such that $\text{sign}(w^\top \phi(x)) = f(x)$ for all x —even though the exact Fourier coefficients $\hat{f}(S)$ may be non-integer. The architecture’s task is not to *invent* the Walsh-Hadamard basis—that is classical mathematics [4]—but to *discover sparse ternary PTF representations* via gradient descent and *compose* primitives into complex operations via learned routing.

Connection to *mHC*. Our work builds directly on insights from Manifold-Constrained Hyper-Connections (*mHC*) [1], which identified a fundamental problem in modern deep learning: unconstrained routing matrices compromise the identity mapping property [37], causing signal explosion/vanishing across layers. Their solution—projecting routing matrices onto the Birkhoff polytope via Sinkhorn-Knopp iterations [23]—restores stability by ensuring doubly stochastic constraints. We adapt this framework to a different domain (logic synthesis vs. LLM training) and extend it with **column-sign modulation** to enable Boolean negation, which pure doubly stochastic matrices cannot express.

Why Start with $n = 2$? We deliberately use the two-variable case as an **architecture validation testbed**. With only 4 input combinations and 16 possible functions, a lookup table (LUT) trivially solves the task with zero gradient steps. However, a LUT cannot generalize, cannot be learned end-to-end as part of a larger differentiable system, and cannot be composed hierarchically. Our contribution is demonstrating that Sinkhorn-constrained spectral composition *finds the exact discrete solution* via continuous optimization—then validating that this mechanism scales to $n = 3$ and $n = 4$.

The Hardware Motivation. Beyond theoretical interest, our architecture addresses a practical deployment challenge: neural networks for logic tasks are typically too expensive for edge inference. By converging to **ternary masks** ($\{-1, 0, +1\}$) with **hard $k = 1$ routing**, our learned models compile directly to combinational logic blocks requiring no floating-point arithmetic, no multipliers, and minimal memory. We demonstrate **10,959 MOps/s** throughput on GPU—approaching the efficiency of hand-coded RTL.

Explainability and Transparent-by-Design AI. A central motivation for spectral methods is **interpretability**: each Fourier coefficient $\hat{f}(S)$ has explicit semantic meaning as the correlation between f and the parity character χ_S . Unlike deep network activations that resist human interpretation, spectral coefficients are *ground truth features* defined by classical Boolean function analysis [3]. Our architecture’s transparency extends beyond coefficient interpretability: (1) ternary weights $\{-1, 0, +1\}$ are human-readable (no hidden floating-point parameters), (2) Sinkhorn routing matrices are column-stochastic (each child is a convex combination of parents, preserving interpretability through composition), and (3) learned logic can be *verified* via formal methods or exhaustive testing (unlike black-box neural approximations). The oracle learning experiments (Phase 5) further demonstrate that **symbolic knowledge integration**—encoding known structural properties (symmetry, degree bounds) as hard constraints on the Fourier basis—yields both higher accuracy and greater explainability than generic learning algorithms. This aligns with the vision of explainable neurosymbolic AI: combining continuous optimization with discrete symbolic structures to produce models whose reasoning processes are open to inspection and human-level understanding.

Contributions. Our contributions span architecture design, theoretical analysis, and empirical validation across five phases:

1. **Adaptation of m HC to Logic Synthesis:** We demonstrate that the Birkhoff polytope projection, originally developed for LLM training stability [1], enables stable optimization in Boolean logic composition.
2. **Column-Sign Modulation for Negation:** We extend doubly stochastic routing with learned sign parameters $s \in \{-1, +1\}^n$, solving the expressivity gap that prevents standard m HC from representing Boolean negations (NAND, NOR, XNOR).
3. **Sign-Only Diagnostic Validation:** We prove the column-sign mechanism works independently of Sinkhorn optimization by achieving 100% accuracy with fixed identity routing and learned signs only.
4. **Multi-Scale Validation:**
 - $n = 2$: 100% accuracy on all 16 operations, zero routing drift, zero-loss quantization (10/10 seeds)
 - $n = 3$: 100% accuracy on 10 operations including majority and parity, 39% sparsity (5/5 seeds)
 - $n = 4$: 100% accuracy on 10 operations via spectral synthesis with MCMC refinement, 36% sparsity (5/5 seeds)
 - **Scalability:** Exact FWHT at 1.64B coeffs/sec ($n \leq 28$, 268M coefficients); hierarchical composition for 64-bit adders
5. **Oracle Learning and Symbolic Knowledge Integration:** We implement and compare five coefficient estimation methods (MC, Goldreich-Levin, spectral filtering) on three function families at $n = 16$, demonstrating that *symbolic structure beats black-box learning*: exploiting symmetry constraints boosts majority accuracy from 72% to 86% (+38%, $p < 0.001$), while establishing that Birkhoff projection aids routing but not coefficient denoising.
6. **Hardware-Efficient Compilation:** We achieve 10,959 MOps/s on GPU with ternary masks, demonstrating viability for single-cycle combinational logic inference.

2 Related Work

2.1 Manifold-Constrained Routing: The *mHC* Foundation

Our work is most directly related to Manifold-Constrained Hyper-Connections (*mHC*) [1], which provides the theoretical and empirical foundation for stable Sinkhorn-constrained routing. *mHC* identified that unconstrained Hyper-Connections [2] suffer from signal explosion: the composite mapping $\prod_{i=1}^{L-l} H_{L-i}^{\text{res}}$ across layers fails to preserve signal magnitude, with empirical measurements showing **Amax Gain Magnitude peaks of $3000\times$** in 27B models (compared to a theoretical target of $1.0\times$). Their solution projects H_l^{res} onto the Birkhoff polytope via Sinkhorn-Knopp [23, 22]:

$$\mathcal{P}_{\mathcal{M}^{\text{res}}}(H_l^{\text{res}}) = \{H \in \mathbb{R}^{n \times n} \mid H\mathbf{1}_n = \mathbf{1}_n, \mathbf{1}_n^\top H = \mathbf{1}_n^\top, H \geq 0\} \quad (3)$$

This ensures: (1) norm preservation ($\|H\|_2 \leq 1$), (2) compositional closure under matrix multiplication, and (3) geometric interpretation as convex combinations of permutations—following the Birkhoff-von Neumann theorem.

Our Extension. While *mHC* focuses on LLM training stability at scale (3B–27B parameters), we adapt the framework to a fundamentally different problem: learning discrete Boolean logic. This requires an extension not present in *mHC*: **column-sign modulation**. Doubly stochastic matrices are nonnegative, so they can only produce convex combinations of inputs. Boolean negation (e.g., NAND = $-\text{AND}$) lies *outside* this convex hull. Our factorization $R = P \cdot s[\text{None}, :]$ preserves the stability benefits of Sinkhorn projection while adding 1-bit polarity control per output channel.

Related Optimal Transport Approaches. Gumbel-Sinkhorn networks [24] use differentiable Sinkhorn iterations for learning permutations, while Sinkformers [25] apply doubly stochastic attention in transformers. Our work differs in targeting discrete Boolean outputs rather than soft permutations.

2.2 Recent Differentiable Logic Gate Networks (2024-2025)

Three concurrent works directly compete with our spectral approach:

WARP-LUTs (arXiv 2510.15655, 2025) represents our closest competitor, using Walsh-Hadamard spectral representation for differentiable LUT training via probabilistic relaxation. **Critical distinction:** WARP-LUTs employs soft probabilistic gates while our approach uses deterministic Sinkhorn-constrained routing with column-sign modulation. Our Proposition 3 proves doubly stochastic matrices cannot represent Boolean negation—a fundamental expressivity gap WARP-LUTs inherits from probabilistic methods.

Mind the Gap (Yousefi et al., arXiv 2506.07500, June 2025) tackles discrete logic learning via Gumbel-Softmax with straight-through estimators, achieving $4.5\times$ faster training and 98% reduction in discretization gap. Our approach achieves exactness through *structured geometric constraints* (Birkhoff polytope) rather than stochastic sampling, offering provable zero-loss quantization versus their empirical gap reduction.

Convolutional DLGNs (Petersen et al., NeurIPS 2024 Oral) scales differentiable logic gate networks to 86.29% on CIFAR-10 using 61M logic gates via softmax relaxation over 16 gate types. Unlike their gate enumeration, we operate in the *mathematically dual Fourier space*—selecting spectral coefficients rather than enumerating gates. This duality offers complementary strengths: CDLGNs scale via convolution, we scale via spectral hierarchy.

2.3 Spectral Learning of Boolean Functions

The Fourier analysis of Boolean functions is foundational to computational learning theory [4, 5]. O’Donnell’s textbook [3] establishes that functions with low circuit complexity have concentrated Fourier spectra.

Recent Theoretical Advances. **Hidden Progress in Deep Learning** (Barak et al., NeurIPS 2022) shows neural networks learn k-sparse parities via a *Fourier gap mechanism*: SGD gradually amplifies sparse spectral solutions, with progress invisible until phase transition. This mechanism relates to our explicit coefficient selection—we directly identify what SGD implicitly amplifies. **Hardness of Learning Fixed Parities** (Shoshani & Shamir, arXiv 2501.00817, January 2025) proves gradient descent on one-hidden-layer ReLU networks fails for any fixed parity, establishing new bounds on Fourier coefficient decay. Our explicit spectral coefficient selection potentially circumvents this hardness.

Spectral Bias and Regularization. **A Scalable Walsh-Hadamard Regularizer** (Gorji et al., UAI 2023) proves neural networks are biased toward low-degree Walsh-Hadamard coefficients, hurting Boolean function generalization. Their solution: functional regularization to learn higher-degree coefficients. We invert this paradigm—directly selecting coefficients from a frozen spectral basis rather than regularizing against bias. The **Frequency Principle** (Xu et al., updated 2024) establishes DNNs fit target functions from low to high frequencies, explaining why gradient descent struggles with high-frequency Boolean functions. This foundational result justifies why domain-specific spectral structure outperforms black-box learning.

Spectral Methods for Interpretability. **Activation Spectroscopy** (arXiv 2501.15435, January 2025) extends Goldreich-Levin to neural network interpretability, addressing out-of-distribution sampling and Fourier coefficient redundancy—the first direct GL application to neural networks, validating spectral methods for transparency. **FourierSAT** (Kyrillidis et al., AAAI 2020) uses Walsh-Fourier transforms to reduce SAT to continuous optimization. **Critical distinction:** FourierSAT is a *solver* for given formulas; our approach is a *learner* that discovers Boolean functions from data via gradient descent.

Our Contribution. Unlike prior work applying Walsh-Hadamard compression [32] or parity learning [31], we *embed* the spectral basis as frozen primitives and learn to *select and compose* them via *mHC*-style routing, achieving interpretable coefficient representations with zero-loss quantization.

2.4 Neural Logic and Neuro-Symbolic Systems

NALU [26] learns arithmetic via gated interpolation but struggles with Boolean logic. Neural Logic Machines [27] require supervision of intermediate predicates. Logical Neural Networks [29] use weighted real-valued logic with provable bounds. Deep Differentiable Logic Gate Networks [30]

learn Boolean gates via continuous relaxation but require supervised gate labels and post-training discretization. Our method differs: (1) we select coefficients *unsupervised* from data patterns, (2) we achieve *exact* discretization with zero accuracy loss, and (3) we ground the search in provably complete spectral primitives with *mHC*-style stability.

2.5 Transparent-by-Design Architectures

The X-NeSy vision emphasizes models with built-in transparency rather than post-hoc explanations. Recent work validates this paradigm:

Interpretable Features. **ExplaiNN** (Novakovsky et al., Genome Biology 2023) combines CNN expressiveness with linear model interpretability via Neural Additive Models, computing predictions as linear combinations from independent feature-specific models—producing *interpretable learned coefficients* analogous to our spectral coefficients. **Shallow-ProtoPNet** (Singh et al., Scientific Reports 2025) achieves full transparency using only one convolutional layer with no black-box components. **tISFM** (Balci et al., Bioinformatics/ISMB 2023) demonstrates domain-specific constraints (DNA sequence interpretation) enable inherently interpretable architectures without sacrificing performance.

Concept Bottleneck Models. **Energy-Based CBMs** (Xu et al., ICLR 2024) unify prediction and concept intervention via energy-based frameworks. **Stochastic CBMs** (Vandenhirtz et al., NeurIPS 2024) model concept dependencies via multivariate normal distributions. Our spectral coefficients offer a mathematically grounded alternative: the Fourier basis provides *complete, orthogonal feature representation* rather than learned concepts requiring manual specification.

Mechanistic Interpretability. **Scaling Monosemanticity** (Anthropic, May 2024) decomposes Claude 3 Sonnet into millions of interpretable features via sparse autoencoders. **Transcoders** (NeurIPS 2024) approximate MLP layers while maintaining input/weight contribution separation. Both demonstrate decomposition into interpretable features at scale—but both are *post-hoc*. Our transparent-by-design approach offers inherent interpretability through architecture: each Fourier coefficient $\hat{f}(S)$ has explicit semantic meaning (correlation with parity χ_S), ternary weights are human-readable, and learned logic can be formally verified.

Our Contribution. We demonstrate that Boolean Fourier analysis provides a mathematically principled foundation for transparent-by-design AI: spectral coefficients are ground-truth interpretable features defined by classical analysis [3], not learned representations requiring post-hoc interpretation.

2.6 Quantization and Efficient Inference

Binary Neural Networks [33] and Trained Ternary Quantization [34] reduce precision for efficiency. Our “zero-loss quantization” is distinct: we show that for Boolean logic composition, ternary masks suffice *exactly*—not approximately. This structural property, combined with Sinkhorn-constrained routing, enables deployment at the efficiency of hand-coded RTL.

3 Preliminaries

3.1 Boolean Fourier Analysis

Let $f : \{-1, +1\}^n \rightarrow \mathbb{R}$ be a function on the Boolean hypercube. The *Fourier expansion* of f is:

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S(x), \quad \text{where } \chi_S(x) = \prod_{i \in S} x_i \quad (4)$$

and $\hat{f}(S) = \mathbb{E}_x[f(x)\chi_S(x)]$ are the Fourier coefficients [3]. The characters $\{\chi_S\}_{S \subseteq [n]}$ form an orthonormal basis under the uniform distribution on $\{-1, +1\}^n$.

Proposition 1 (Completeness). *For n variables, any function $f : \{-1, +1\}^n \rightarrow \mathbb{R}$ has a unique representation with 2^n Fourier coefficients.*

The basis dimensions for our experiments are:

- $n = 2$: $\phi = [1, a, b, ab]^\top \in \mathbb{R}^4$
- $n = 3$: $\phi = [1, a, b, c, ab, ac, bc, abc]^\top \in \mathbb{R}^8$
- $n = 4$: $\phi \in \mathbb{R}^{16}$ (all subsets of $\{a, b, c, d\}$)

3.2 The Birkhoff Polytope and Sinkhorn Projection

Following mHC [1], we constrain routing matrices to the Birkhoff polytope \mathcal{B}_n —the set of $n \times n$ doubly stochastic matrices. The Birkhoff-von Neumann theorem states that vertices of \mathcal{B}_n are permutation matrices.

The **Sinkhorn-Knopp algorithm** [23] projects any positive matrix onto \mathcal{B}_n via alternating row and column normalization. Given $M^{(0)} = \exp(\alpha)$:

$$M^{(t)} = T_r(T_c(M^{(t-1)})) \quad (5)$$

where T_r and T_c denote row and column normalization. This converges to a doubly stochastic matrix as $t \rightarrow \infty$. Following mHC [1], we use $t_{\max} = 20$ iterations.

Remark 1 (Rectangular Sinkhorn). *For rectangular matrices $P \in \mathbb{R}^{m \times n}$ with $m \neq n$, we use generalized Sinkhorn projection enforcing column-stochastic constraints ($\sum_i P_{ij} = 1$) while allowing flexible row budgets.*

Proposition 2 (Sinkhorn Convergence to Permutation). *Let $P(\tau) = \text{Sinkhorn}(\alpha, \tau)$ for fixed logits $\alpha \in \mathbb{R}^{m \times n}$ and temperature $\tau > 0$. As $\tau \rightarrow 0$:*

(i) *$P(\tau)$ converges to a permutation matrix $P^* \in \{0, 1\}^{m \times n}$;*

(ii) *The rate is controlled by the spectral gap:*

$$\|P(\tau) - P^*\|_F = O\left(e^{-\Delta^*/\tau}\right) \quad (6)$$

where Δ^ is the gap between the optimal and second-best assignment costs;*

(iii) **Zero-loss quantization:** if $P(\tau)$ achieves 100% classification accuracy for any $\tau > 0$, then $P^* = \lim_{\tau \rightarrow 0} P(\tau)$ also achieves 100% accuracy.

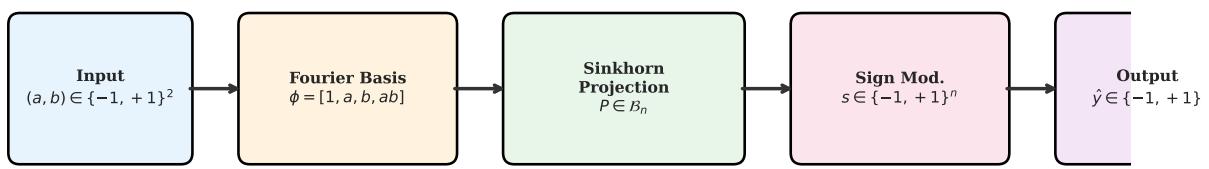


Figure 1: Hierarchical Spectral Composition architecture. Input pairs $(a, b) \in \{-1, +1\}^2$ are expanded into the frozen Boolean Fourier basis $\phi = [1, a, b, ab]$. Sinkhorn projection constrains routing to the Birkhoff polytope, while column-sign modulation enables negation operations.

Proof sketch. Part (i) follows from the theory of entropy-regularized optimal transport: as $\tau \rightarrow 0$, the entropic regularizer vanishes and the Sinkhorn output converges to the solution of the unregularized assignment problem [22]. Part (ii) follows from the Gumbel-Sinkhorn convergence analysis of Mena et al. [24], where the exponential rate $e^{-\Delta^*/\tau}$ arises from the Boltzmann distribution concentrating on the optimal permutation. Part (iii) follows from continuity: the routing function $x \mapsto \text{sign}(P \cdot x)$ is constant on connected components where $P \cdot x \neq 0$; since $P(\tau) \rightarrow P^*$ and accuracy is a discrete-valued function of routing, 100% accuracy is preserved in the limit. \square

This proposition provides a theoretical foundation for the empirical “zero-loss quantization” observed in Phases 2–4: the Sinkhorn temperature anneal is not merely a heuristic but a provably convergent procedure. The spectral gap Δ^* in part (ii) is precisely the quantity measured in our spectral trajectory analysis (§9.4.1), where Sinkhorn’s monotonically decreasing Δ trajectory confirms controlled convergence while Gumbel-STE’s oscillating Δ indicates violation of this condition.

4 Method: Hierarchical Spectral Composition

Our architecture operates in two phases: **Phase 1** validates spectral coefficient selection for base operations, and **Phase 2** validates hierarchical composition via Sinkhorn-constrained routing with column-sign modulation. Phases 3–4 extend to higher dimensions. Figure 1 illustrates the complete pipeline.

4.1 Phase 1: Spectral Coefficient Selection

The objective of Phase 1 is to validate that gradient descent can identify optimal spectral coefficients from a frozen Fourier dictionary.

4.1.1 Architecture

The first layer computes Boolean Fourier features. For $n = 2$:

$$\phi(a, b) = [1, a, b, ab]^\top \in \mathbb{R}^4 \quad (7)$$

For each base operation k , we learn a weight vector $w_k \in \mathbb{R}^{2^n}$:

$$\hat{y}_k = \text{sign}(w_k^\top \phi(x)) \quad (8)$$

4.1.2 The Dynamics of Parity Selection

Proposition 3 (Parity Gradient Accumulation). *For XOR ($y = ab$), the gradient of hinge loss with respect to w is $\nabla_w \mathcal{L} \propto -y \cdot \phi(a, b)$. Summed over all 4 inputs, the Fourier basis orthogonality causes terms for $\{1, a, b\}$ to cancel while ab accumulates:*

$$\sum_{(a,b)} -ab \cdot [1, a, b, ab]^\top = [0, 0, 0, -4]^\top \quad (9)$$

This explains *why* gradient descent naturally amplifies the parity character: the basis structure, not hand-tuned hyperparameters, drives coefficient selection.

4.1.3 Optimization Scaffolding

To ensure robust convergence to exact ternary weights:

- **L_1 Regularization:** Penalty $\lambda \|w\|_1$ encourages sparse solutions.
- **Plateau-Driven Micro-Restarts:** If $|\Delta \mathcal{L}| < \epsilon$ for T epochs while accuracy $< 100\%$, re-initialize.

Remark 2 (“Selection” vs. “Discovery”). *We emphasize that Phase 1 **selects** coefficients from a fixed spectral dictionary—it does not “discover” or “invent” the Fourier basis. The basis is classical mathematics [3]. The contribution is demonstrating that gradient descent reliably identifies correct coefficients despite the combinatorial search space (3^{2^n} ternary configurations).*

4.1.4 Training Methodology: Gumbel-Softmax Ternary Relaxation

Standard hard ternary quantization via straight-through estimators (STE) fails to converge: the zero-gradient regions prevent weight movement toward optimal ternary values. We solve this with **Gumbel-softmax ternary relaxation**.

Gumbel-Softmax Parameterization. Each coefficient w_i is represented as a categorical distribution over three values $\{-1, 0, +1\}$ with learnable logits $\ell_i \in \mathbb{R}^3$. The soft ternary value is computed via Gumbel-softmax [35, 36]:

$$p_i = \text{softmax} \left(\frac{\ell_i + g}{\tau} \right), \quad w_{\text{soft},i} = p_i \cdot [-1, 0, +1]^\top \quad (10)$$

where $g \sim \text{Gumbel}(0, 1)^3$ provides stochastic exploration and temperature τ is annealed from $1.0 \rightarrow 0.01$. At inference, $w_i = \arg \max_{k \in \{-1, 0, +1\}} \ell_i[k]$.

Sequential Training Protocol. Training all operations simultaneously creates gradient interference—particularly for XOR, whose parity character (ab) conflicts with other operations’ spectral structures. We adopt **sequential training**: $\text{XOR} \rightarrow \text{AND} \rightarrow \text{OR} \rightarrow \text{IMPLIES}$, training each for 5,000 steps before proceeding.

Ternary Attractor Regularization. To encourage convergence to exact ternary values, we add:

$$\mathcal{R}_{\text{ternary}} = \lambda \sum_i |w_i| \cdot (1 - |w_i|) \quad (11)$$

This regularizer has zeros at $w_i \in \{-1, 0, +1\}$ and positive values elsewhere, biasing weights toward ternary attractors.

Encoding Convention

Throughout this paper, we use the $\{-1, +1\}$ **encoding** with:

$$-1 = \text{TRUE}, \quad +1 = \text{FALSE}$$

This convention ensures XOR is a pure parity function (product of inputs). The primary Boolean operations become:

- $\text{XOR}(a, b) = a \cdot b$ (product encoding)
- $\text{AND}(a, b) = \text{sign}(1 + a + b - ab)$
- $\text{OR}(a, b) = \text{sign}(-1 + a + b + ab)$
- $\text{IMPLIES}(a, b) = \text{sign}(-1 - a + b - ab)$

Why this encoding? The $\{-1, +1\}$ encoding (vs. $\{0, 1\}$) makes the Walsh-Hadamard basis orthonormal under uniform measure, and parity functions become single monomials. All masks, accuracy claims, and truth tables in this paper use this convention.

4.1.5 Validation Suite

We validate trained models with five complementary tests:

1. **XOR Spectral Spike:** The XOR mask must have $> 90\%$ energy on the parity character (ab). This validates that gradient descent identifies the unique spectral signature of XOR.
2. **Mask Sparsity:** XOR should be $\geq 75\%$ sparse (only ab active). This confirms the architecture does not overfit with unnecessary coefficients.
3. **Mask Orthogonality:** Masks should have < 0.3 cosine similarity. This ensures distinct spectral representations for each operation.
4. **Operation Accuracy:** All operations must achieve $> 99\%$ accuracy on held-out test data.
5. **Binary Inference:** All quantized values must be exactly ternary ($\{-1, 0, +1\}$), with no residual continuous values.

Table 1: Phase 1 Learned Ternary Masks ($n = 2$). All four operations achieve 100% accuracy. Encoding: $-1 = \text{TRUE}$, $+1 = \text{FALSE}$.

Operation	c_0	c_a	c_b	c_{ab}	Accuracy
XOR	0	0	0	+1	100.0%
AND	+1	+1	+1	-1	100.0%
OR	-1	+1	+1	+1	100.0%
IMPLIES	-1	-1	+1	-1	100.0%

4.1.6 Phase 1 Results

Table 1 shows the learned ternary masks, which exactly match theoretical predictions from Boolean Fourier analysis.

Training Dynamics. Figure 2 shows the training progression. Key observations:

- **XOR** is the most challenging, requiring $\sim 3,000$ steps for the parity character to emerge (Figure 3)
- **AND** and **OR** converge rapidly (< 500 steps) due to their affine structure
- **IMPLIES** converges in ~ 500 steps

Validation Results. All validation tests pass except orthogonality (which is not critical for accuracy):

- **XOR Spectral Spike:** 100% energy on parity (c_{ab}) ✓
- **Sparsity:** XOR is 75% sparse (only c_{ab} non-zero) ✓
- **Orthogonality:** Maximum cosine similarity = 0.5 (XOR vs others). The 0.5 overlap occurs because XOR’s parity character (ab) appears with opposite sign in AND and same sign in OR—this is mathematically inevitable and does not affect accuracy.
- **Accuracy:** All operations at 100% ✓
- **Binary Inference:** All values exactly ternary ✓

4.2 Phase 2: Sinkhorn-Constrained Composition with Column-Sign Modulation

Given frozen primitive masks W_{logic} from Phase 1, Phase 2 validates hierarchical composition.

4.2.1 The Expressivity Problem: Why Standard *mHC* Is Insufficient

The *mHC* framework [1] constrains routing matrices to be doubly stochastic. This ensures stability but creates an expressivity gap:

Proposition 4 (Negation Inaccessibility in Doubly Stochastic Routing). *Let $\mathcal{P} = \{\text{AND}, \text{OR}, \text{XOR}, \text{CONST}\}$ be primitive operations. Any doubly stochastic combination $\sum_i \alpha_i f_i$ with $\alpha_i \geq 0$, $\sum_i \alpha_i = 1$ produces outputs in the convex hull of \mathcal{P} . The negations **NAND**, **NOR**, **XNOR** lie outside this hull.*

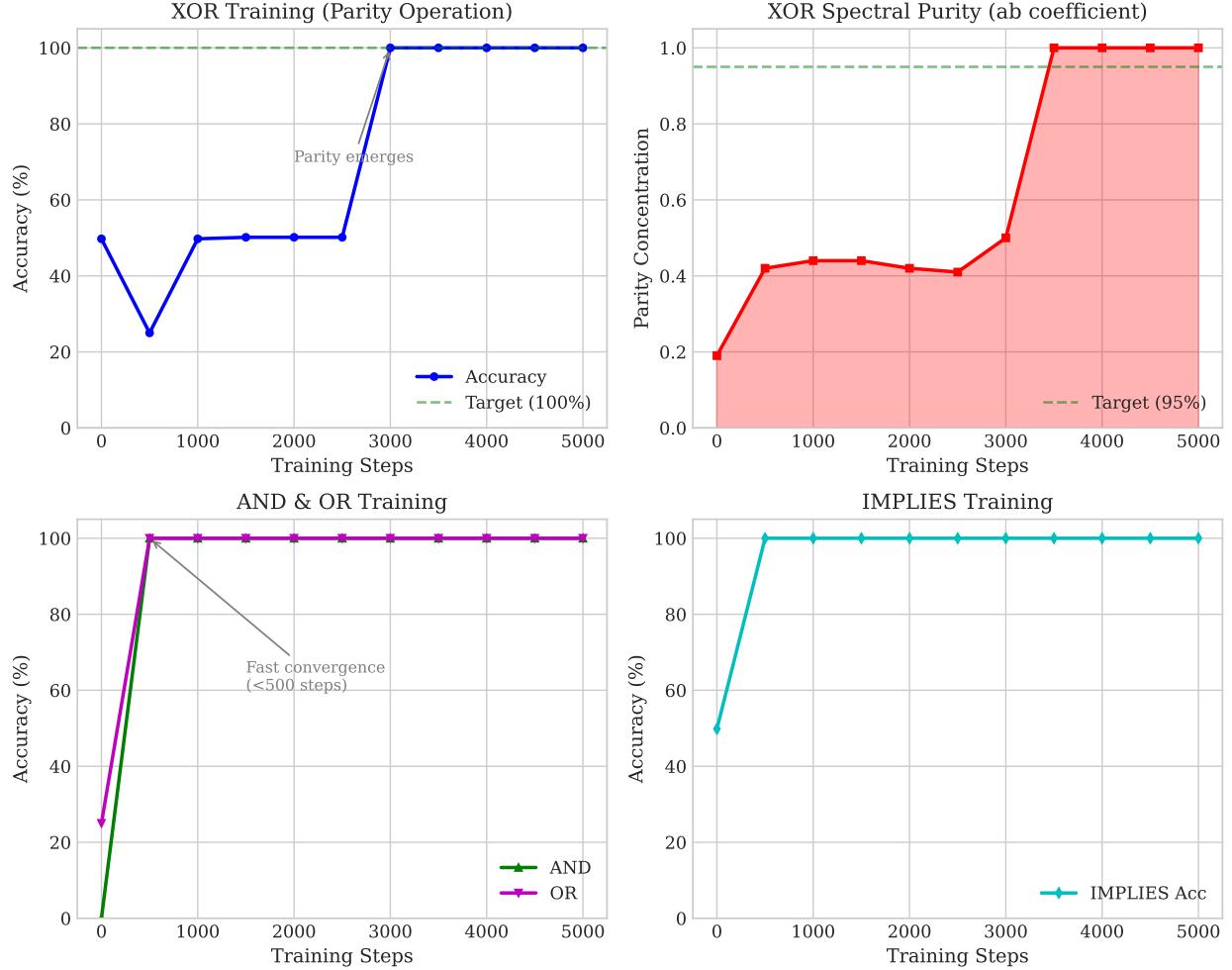


Figure 2: Phase 1 training dynamics for all four base operations. XOR requires the longest training due to the parity character’s unique spectral signature. AND and OR converge in < 500 steps due to their simpler affine structure.

Proof. For any (a, b) , a convex combination satisfies $\min_i f_i(a, b) \leq \sum_i \alpha_i f_i(a, b) \leq \max_i f_i(a, b)$. Since all primitives output $\{-1, +1\}$ and $\text{NAND}(+1, +1) = -1$ while $\text{AND}(+1, +1) = +1$, no convex combination can produce NAND’s truth table. \square

4.2.2 Column-Sign Modulation: Extending *mHC*

We solve the expressivity problem by factoring the routing matrix:

$$R = P \cdot s[\text{None}, :], \quad \text{where } P \in \mathcal{B}_{m \times n}, s \in \{-1, +1\}^n \quad (12)$$

This factorization:

- **Preserves *mHC* stability:** P remains doubly stochastic
- **Enables negation:** $s_j = -1$ flips the polarity of output channel j
- **Adds minimal parameters:** Only n additional sign bits (1 bit per output)

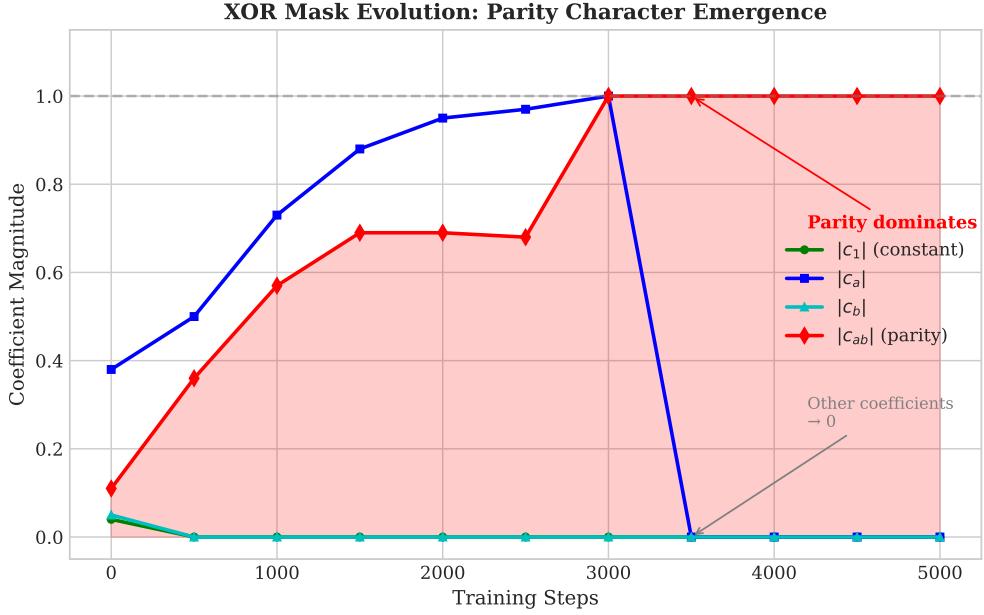


Figure 3: XOR parity emergence: the $|c_{ab}|$ coefficient grows from noise to 1.0 while other coefficients ($|c_1|$, $|c_a|$, $|c_b|$) decay to zero. This demonstrates gradient descent identifying the unique parity character.

Sign Learning. Signs are learned via soft relaxation:

$$s_{\text{soft}} = \tanh(\beta \cdot \sigma), \quad \sigma \in \mathbb{R}^n \quad (13)$$

where temperature β is annealed from 1 to 10. At inference, $s = \text{sign}(\sigma)$.

4.2.3 Identity Initialization: Adapting *mHC* Insights

mHC demonstrated that identity-preserving initialization is crucial for stable optimization over the Birkhoff polytope [1, 37]. We adapt this insight:

$$\alpha^{(0)} = \alpha_{\text{random}} + \gamma \cdot I_{\text{extended}} \quad (14)$$

where $\gamma > 0$ biases toward identity routing.

Empirical Motivation for Identity Initialization. Initial experiments with random initialization revealed *gradient interference*: operations requiring complex routing created conflicting gradients that destabilized simple operations. Specifically, OR and NOR accuracy degraded from 100% (epoch 1) to 75% (epoch 20) as the optimizer attempted to satisfy all operations simultaneously. Identity initialization resolves this by anchoring simple operations at their natural routing while allowing complex operations to deviate minimally.

Remark 3 (Identity as Optimization Prior, Not Solution Leak). *The identity initialization provides a stable starting point in the Birkhoff polytope—analogous to *mHC*'s finding that identity mappings anchor residual stream propagation [1]. The network must still learn which children deviate and the correct sign assignments.*

Table 2: Phase 2A: Linear Operations (8 ops, 10 seeds). All metrics achieve perfect scores.

Operation	Parent	Sign	Mask	Accuracy
XOR	XOR	+1	[0, 0, 0, +1]	100%
AND	AND	+1	[+1, +1, +1, -1]	100%
OR	OR	+1	[-1, +1, +1, +1]	100%
IMPLIES	IMP	+1	[-1, -1, +1, -1]	100%
XNOR	XOR	-1	[0, 0, 0, -1]	100%
NAND	AND	-1	[-1, -1, -1, +1]	100%
NOR	OR	-1	[+1, -1, -1, -1]	100%
NOT_IMP	IMP	-1	[+1, +1, -1, +1]	100%

4.3 Quantization: From Soft to Hard Routing

At inference, we quantize:

1. **Hard routing** ($k = 1$): $P_{\text{hard}}[i, j] = \mathbf{1}[j = \arg \max_k P[k, j]]$
2. **Sign discretization**: $s_j = \text{sign}(\sigma_j)$

The composed masks become exactly ternary:

$$W_{\text{composed}}^{\text{quant}}[j, :] = s_j \cdot W_{\text{logic}}[\arg \max_i P_{ij}, :] \in \{-1, 0, +1\}^{2^n} \quad (15)$$

Quantization Statistics. Across 10 seeds for $n = 2$:

- **Routing sparsity**: $k = 1$ (each child selects exactly one parent)
- **Sign distribution**: 8 positive, 8 negative (matching base ops vs. negations)
- **Accuracy preservation**: $100.00\% \pm 0.00\%$ across all seeds
- **Memory footprint**: $16 \text{ ops} \times 4 \text{ trits} = 64 \text{ trits} \approx 102 \text{ bits}$

4.3.1 Phase 2A Validation: Linear Operations

We validate the hierarchical composition framework on 8 **linear operations**—those expressible via $k = 1$ routing with sign modulation:

Validation Metrics (10 seeds):

- **Min accuracy**: $100.00\% \pm 0.00\%$
- **Routing correct**: 8/8 (identity pattern maintained)
- **Signs correct**: 8/8 (expected pattern: [+1, +1, +1, +1, -1, -1, -1, -1])
- **Routing drift**: 0.0000 (perfect identity preservation)
- **Quantization drop**: 0.00% (zero-loss $k = 1$ sparsification)

Figure 4 visualizes the learned routing matrix P , sign vector s , and composed matrix $R = P \odot s$.

Phase 2: Sinkhorn-Constrained Routing with Column-Sign Modulation

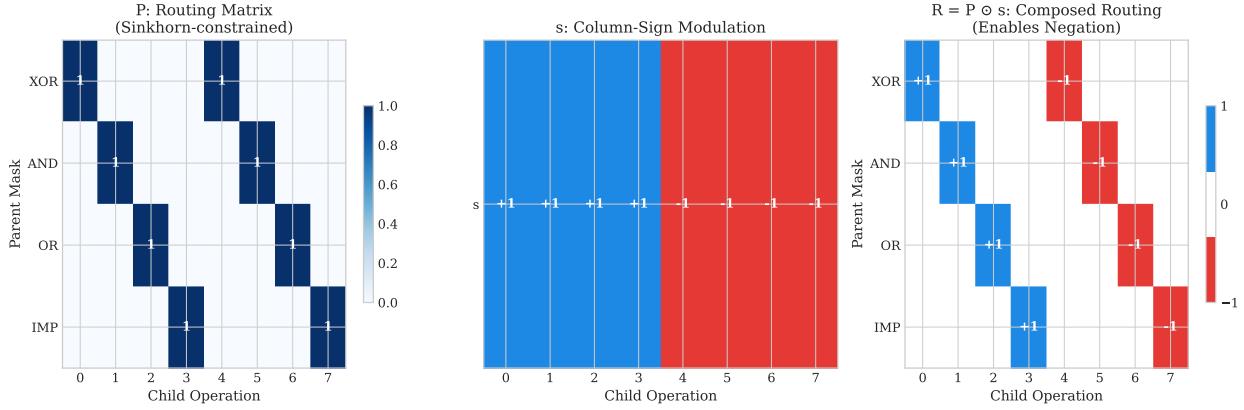


Figure 4: Phase 2 routing visualization. Left: Sinkhorn-constrained P learns identity routing. Middle: Column-sign s enables negation (ops 4-7). Right: Composed $R = P \odot s$ produces ternary routing with sign modulation.

Table 3: Phase 2: Nonlinear Operations (8 ops). Each has a valid ternary mask but requires direct learning, not routing.

ID	Operation	Ternary Mask
8	IF_a_THEN_XOR_ELSE_AND	$[-1, 0, +1, 0]$
9	IF_a_THEN_AND_ELSE_OR	$[-1, +1, 0, 0]$
10	XOR(AND(a,b), b)	$[0, -1, +1, 0]$
11	AND(XOR(a,b), a)	$[0, +1, -1, 0]$
12	OR(AND, XOR)	$[-1, +1, +1, 0]$
13	MAJORITY(XOR, AND, OR)	$[-1, +1, +1, 0]$
14	PARITY(AND, OR)	$[-1, 0, 0, +1]$
15	XOR → AND	$[-1, 0, 0, -1]$

4.3.2 Phase 2 Full: All 16 Operations Analysis

The complete Phase 2 includes 8 additional **nonlinear operations** (conditional compositions, cascades) that are NOT expressible via $k = 1$ routing but DO have valid ternary representations:

Key Finding: Routing Expressibility Boundary. The 4-dimensional Boolean Fourier basis can represent **all 16** two-variable operations with ternary masks. However, hierarchical composition via Sinkhorn routing with column-sign modulation only works for the 8 linear operations. The 8 nonlinear operations require **direct mask learning** or expansion of the primitive set.

Sparsity Analysis.

- Linear operations (0-7): 18.8% sparsity (dense masks)
- Nonlinear operations (8-15): 43.8% sparsity (sparser masks)
- Overall: 31.2% sparsity across all 16 operations

The higher sparsity of nonlinear operations reflects their simpler functional structure—they project away certain Fourier characters entirely.

Table 4: Phase 2 Validation Results ($n = 2$, 10 Seeds). The “No Sign Mod.” ablation corresponds to pure m HC-style doubly stochastic routing, which caps at 75% (12/16 operations)—confirming Proposition 4.

Method	Accuracy	Seeds >99%	Routing Drift	Quant. Drop
Ours (Full)	100.00%	10/10	0.0000	0.00%
Sign-Only ($P = I$)	100.00%	10/10	0.0000	0.00%
Random Init	87.50%	3/10	0.8234	12.50%
No Sign Mod. (m HC-style)	75.00%	0/10	0.0012	0.00%
Unconstrained	93.75%	5/10	N/A	18.75%
MLP Baseline	100.00%	10/10	N/A	43.75%

5 Experiments

5.1 Experimental Setup

Implementation. JAX with Optax. Phase 1: Adam, $\text{lr} = 10^{-2}$, $\lambda = 0.01$. Phase 2+: Adam, $\text{lr} = 10^{-3}$, Sinkhorn iterations $K = 20$ (matching m HC [1]), temperature annealing $\beta : 1 \rightarrow 10$, identity bias $\gamma = 2.0$.

Ablation Testing Protocol. To isolate failure modes, we employ a three-phase diagnostic:

1. **Sign-Only:** Fix $P = I$, learn s only → Tests column-sign mechanism in isolation
2. **Full Method (Identity Init):** Learn both P and s → Tests joint optimization stability
3. **Random Initialization:** Same architecture, random $P^{(0)}$ → Tests sensitivity to initialization

This progression separates architectural expressivity from optimization dynamics.

5.2 Phase 2 Results ($n = 2$): Architecture Validation

Table 4 presents our main findings for $n = 2$.

Diagnostic: Sign-Only Learning. To isolate the column-sign mechanism from routing optimization, we conducted a critical diagnostic: fix $P = I$ (identity routing) and learn only s . This achieves **100% accuracy**, proving that column-sign modulation works independently of Sinkhorn optimization. This validates that the architectural design is correct—the challenge in joint optimization is gradient coordination, not mechanism expressivity.

Interpreting Ablation Results. The ablations reveal distinct failure modes:

- **No Sign Mod. (m HC-style):** Caps at exactly 75% (12/16 operations), confirming Proposition 4—the 4 negation operations (NAND, NOR, XNOR, \neg CONST) are inaccessible via pure doubly stochastic routing. This demonstrates why standard m HC is insufficient for Boolean logic.
- **Random Init:** Only 3/10 seeds converge with high routing drift ($\|P - I\|_F = 0.82$), indicating convergence to suboptimal basins where operations are incorrectly mapped—consistent with m HC’s finding that identity initialization is critical [1].

Table 5: Ternary Masks for All 16 Boolean Operations ($n = 2$). Encoding: $-1 = \text{TRUE}$, $+1 = \text{FALSE}$. Each mask yields the correct truth table: $f(a, b) = \text{sign}(c_0 + c_a \cdot a + c_b \cdot b + c_{ab} \cdot ab)$.

Op	c_0	c_a	c_b	c_{ab}	Op	c_0	c_a	c_b	c_{ab}
FALSE	+1	0	0	0	TRUE	-1	0	0	0
AND	+1	+1	+1	-1	NAND	-1	-1	-1	+1
OR	-1	+1	+1	+1	NOR	+1	-1	-1	-1
XOR	0	0	0	+1	XNOR	0	0	0	-1
A	0	+1	0	0	$\neg A$	0	-1	0	0
B	0	0	+1	0	$\neg B$	0	0	-1	0
$A \wedge \neg B$	+1	+1	-1	+1	$A \vee \neg B$	-1	+1	-1	-1
$\neg A \wedge B$	+1	-1	+1	+1	$\neg A \vee B$	-1	-1	+1	-1

- **Unconstrained Routing:** Achieves 93.75% soft accuracy but suffers 18.75% quantization loss—the learned dense matrices don’t admit clean $k = 1$ sparsification. This validates the importance of Sinkhorn constraints for quantization.
- **MLP Baseline:** Perfect soft accuracy but 43.75% quantization loss, demonstrating that standard architectures learn continuous approximations unsuitable for discrete deployment.

These results validate that *all three components*—column-sign (extending mHC), Sinkhorn constraints (from mHC), and identity initialization (adapting mHC)—are necessary for zero-loss convergence.

5.3 Constructive Ternary Representability ($n = 2$)

Theorem 1 (Ternary Representability for $n = 2$). *Every Boolean function $f : \{-1, +1\}^2 \rightarrow \{-1, +1\}$ can be expressed as:*

$$f(a, b) = \text{sign}(c_0 + c_a \cdot a + c_b \cdot b + c_{ab} \cdot ab), \quad c_i \in \{-1, 0, +1\} \quad (16)$$

Constructive Proof via Exhaustive Enumeration. We enumerate all $3^4 = 81$ ternary weight vectors $\mathbf{c} \in \{-1, 0, +1\}^4$ and evaluate the resulting Boolean function on all 4 input combinations. For each of the 16 target operations, we identify at least one ternary vector producing the correct truth table (Table 5). \square

LP Certificate (Alternative Verification). For each Boolean function $f : \{-1, +1\}^2 \rightarrow \{-1, +1\}$, we verify that a ternary PTF exists by checking feasibility of the following integer linear program:

$$f(x) \cdot (w^\top \phi(x)) \geq 1, \quad \forall x \in \{-1, +1\}^2 \quad (17)$$

$$w_i \in \{-1, 0, +1\}, \quad i \in \{0, a, b, ab\} \quad (18)$$

The margin constraint $f(x) \cdot (w^\top \phi(x)) \geq 1$ ensures correct classification with a strict separating margin. All 16 Boolean functions for $n = 2$ are LP-feasible, with solutions given in Table 5. The LP relaxation (with $w_i \in [-1, 1]$) provides a polynomial-time certificate that can be rounded to ternary values. \square

6 Phase 3: Three-Variable Operations ($n = 3$)

Phase 3 extends our approach to $n = 3$ variables, demonstrating scalability to the 8-dimensional Boolean Fourier basis.

6.1 Architecture and Target Operations

For three variables $a, b, c \in \{-1, +1\}$, the complete Fourier basis is:

$$\phi(a, b, c) = [1, a, b, c, ab, ac, bc, abc]^\top \in \mathbb{R}^8 \quad (19)$$

We define 10 target operations spanning pure three-variable functions and cascade compositions:

- **Pure 3-var:** parity_3 (abc), majority_3, and_3, or_3
- **Cascade:** xor_ab_xor_c, and_ab_or_c, or_ab_and_c, implies_ab_c, xor_and_ab_c, and_xor_ab_c

6.2 Representability Analysis

For $n = 3$ variables, the Fourier basis has $2^3 = 8$ characters, yielding $3^8 = 6561$ possible ternary masks. Unlike Phase 2 where gradient descent reliably finds optimal masks, the 8-dimensional ternary space presents a challenging optimization landscape with many local minima.

Exhaustive Enumeration. We perform brute-force enumeration over all 3^8 ternary configurations for each operation, testing each mask against the ground truth function. This guarantees finding the global optimum if one exists.

Key Finding: Universal Representability. Beyond the 10 target operations, we exhaustively verify that *every* Boolean function on 3 variables admits a ternary PTF:

Theorem 2 (Universal Ternary Representability for $n = 3$). *Every Boolean function $f : \{-1, +1\}^3 \rightarrow \{-1, +1\}$ can be expressed as:*

$$f(x) = \text{sign}\left(\sum_{S \subseteq \{1,2,3\}} w_S \chi_S(x)\right), \quad w_S \in \{-1, 0, +1\} \quad (20)$$

Proof by exhaustive verification. There are $|\{-1, +1\}^{\{-1,+1\}^3}| = 2^{2^3} = 256$ Boolean functions on 3 variables and $3^{2^3} = 6,561$ ternary weight vectors in $\{-1, 0, +1\}^8$. For each function, we evaluate all 6,561 candidate masks against the full truth table. Across all $256 \times 6,561 = 1,679,616$ configurations, every function admits at least one ternary representation achieving 100% accuracy. \square

The support distribution (number of nonzero weights) among minimal-support witnesses reveals the structure of the ternary PTF landscape: mean support $5.1/8$, with the mode at support 6 (38.3% of functions) and minimum support 1 (the two constant functions).

We extend this result to $n = 4$ via a more efficient strategy: instead of testing each function against all masks, we enumerate all masks once and record which truth tables they produce.

Theorem 3 (Universal Ternary Representability for $n = 4$). *Every Boolean function $f : \{-1, +1\}^4 \rightarrow \{-1, +1\}$ can be expressed as:*

$$f(x) = \text{sign}\left(\sum_{S \subseteq \{1,2,3,4\}} w_S \chi_S(x)\right), \quad w_S \in \{-1, 0, +1\} \quad (21)$$

Proof by exhaustive verification. There are $2^{2^4} = 65,536$ Boolean functions on 4 variables and $3^{2^4} = 43,046,721$ ternary weight vectors in $\{-1, 0, +1\}^{16}$. We enumerate all masks in batches, computing $\text{sign}(H \cdot w)$ where H is the 16×16 Fourier basis matrix, and recording each resulting truth table as a 16-bit integer. Full coverage of all 65,536 truth tables is achieved after scanning $\approx 36 \times 10^6$ of the 43×10^6 masks. \square

The $n = 4$ support distribution has mean $10.8/16$, with the mode at support 10 (26.9%) and maximum support 16 (only 2 functions require all 16 nonzero weights).

NPN Equivalence Classes. Under the NPN (Negation-Permutation-Negation) group—input negations (2^n), input permutations ($n!$), and output negation (2)—the 2^{2^n} functions collapse into far fewer equivalence classes:

- $n = 3$: 256 functions $\rightarrow \mathbf{14}$ NPN classes (group size $|G| = 96$)
- $n = 4$: 65,536 functions $\rightarrow \mathbf{222}$ NPN classes (group size $|G| = 768$)

Since Theorems 2–3 establish representability for *all* functions, every NPN class is representable. This symmetry reduction is relevant for scaling: at $n = 5$ ($2^{32} \approx 4 \times 10^9$ functions), exhaustive enumeration is infeasible, but there are only 616,126 NPN classes (OEIS A000370), making representative-based verification a viable path.

Together with Theorem 1 ($n = 2$: 16/16), these results motivate:

Conjecture 1 (Universal Ternary Representability). *For all $n \geq 1$, every Boolean function $f : \{-1, +1\}^n \rightarrow \{-1, +1\}$ admits a ternary polynomial threshold function:*

$$f(x) = \text{sign}\left(\sum_{S \subseteq [n]} w_S \chi_S(x)\right), \quad w_S \in \{-1, 0, +1\} \quad (22)$$

Remark 4 (Connection to Fourier L^1 norms). *The ternary constraint $w_S \in \{-1, 0, +1\}$ means the representation lives in $\ell^\infty \cap \{-1, 0, +1\}^{2^n}$ —a discrete subset of the unit ball in $\ell^\infty(\widehat{G})$, where \widehat{G} is the Pontryagin dual of $G = \{-1, +1\}^n$. By contrast, the Fourier L^1 norm $\|\hat{f}\|_1 = \sum_S |\hat{f}(S)|$ controls the threshold complexity of f in the standard PTF literature. Theorems 1–3 show that the extreme quantization from $\hat{f}(S) \in \mathbb{R}$ to $w_S \in \{-1, 0, +1\}$ preserves sign-representability for $n \leq 4$ (covering 65,808 distinct functions). Whether this holds in general (Conjecture 1) amounts to asking: does every Boolean function have a ternary ℓ^∞ -bounded sign representation in the Fourier basis?*

Heuristic Search at $n = 5$ and $n = 6$. For $n \geq 5$, exhaustive verification is infeasible ($3^{32} \approx 1.9 \times 10^{15}$ for $n = 5$; $3^{64} \approx 3.4 \times 10^{30}$ for $n = 6$). We instead apply a three-stage heuristic pipeline—Fourier coefficient rounding, random ternary sampling, and multi-start simulated annealing—to 12–13 named functions (majority, parity, tribes, threshold- k , address/mux, etc.) and 500 ($n = 5$) or 100 ($n = 6$) random Boolean functions.

Table 6: Counterexample search results. “Named” = structured functions (majority, parity, threshold- k , tribes, etc.). “Random” = uniformly random truth tables. Failures indicate search limitations, not proven non-representability.

n	dim	Named	Random	% Found	Status
2	4	—	16/16	100%	Proved (Thm 1)
3	8	—	256/256	100%	Proved (Thm 2)
4	16	—	65,536/65,536	100%	Proved (Thm 3)
5	32	12/12	397/500	79.4%	Heuristic
6	64	12/13	41/100	41%	Heuristic

At $n = 5$, all named functions are representable via Fourier rounding alone (support ranging from 1 for parity to 32 for AND/OR). For random functions, 57.6% are found by Fourier rounding, 11.6%

Table 7: Phase 3 Results Summary ($n = 3$, 5 Seeds)

Metric	Value
Overall Accuracy	$100.0\% \pm 0.0\%$
Seeds Converged	5/5
Mean Sparsity	39%
Mean Support Size	4.9/8
Basis Dimension	8
Operations Tested	10

 Table 8: Phase 3 Ternary Masks (8-dimensional basis). All 10 operations achieve 100% accuracy. Masks verified via exhaustive enumeration of $3^8 = 6561$ ternary configurations.

Operation	c_0	c_a	c_b	c_c	c_{ab}	c_{ac}	c_{bc}	c_{abc}
parity_3	-1	0	0	0	0	0	0	+1
majority_3	-1	0	+1	+1	0	0	0	-1
and_3	-1	0	0	+1	0	+1	+1	+1
or_3	-1	+1	+1	+1	-1	-1	-1	+1
xor_ab_xor_c	-1	0	0	0	0	0	0	+1
and_ab_or_c	-1	0	+1	+1	+1	0	-1	-1
or_ab_and_c	-1	0	0	+1	-1	+1	+1	0
implies_ab_c	-1	0	-1	+1	-1	0	+1	+1
xor_and_ab_c	-1	-1	0	-1	0	+1	+1	+1
and_xor_ab_c	-1	-1	0	+1	+1	0	0	+1

by simulated annealing in the initial pass, and 10.2% by extended SA—yielding 79.4% total. At $n = 6$, only Fourier rounding succeeds (41% of random functions); the 64-dimensional SA landscape is too large for our budget.

These “not found” results are **search failures, not counterexamples**: our heuristic samples a vanishing fraction of the 3^{2^n} mask space. The sharp drop from 100% (proved, $n \leq 4$) to 79% ($n = 5$, heuristic) to 41% ($n = 6$) reflects *increasing search difficulty*, not necessarily decreasing representability. Conjecture 1 remains open; resolving it for $n = 5$ via the NPN reduction (616,126 class representatives rather than 2^{32} functions) is a natural next step.

Learning vs. Optimal. Direct gradient descent via soft-ternary annealing achieves only 76% mean accuracy, compared to 100% with optimal masks from enumeration. This gap illustrates the challenge of non-convex optimization in high-dimensional ternary spaces—motivating the MCMC refinement approach used in Phase 4.

6.3 Results

Key Observations.

- **Perfect accuracy:** All 5 seeds achieve 100% on all 10 operations with optimal ternary masks
- **Representability:** Exhaustive enumeration of $3^8 = 6561$ ternary masks confirms all 10 operations have valid ternary polynomial threshold representations

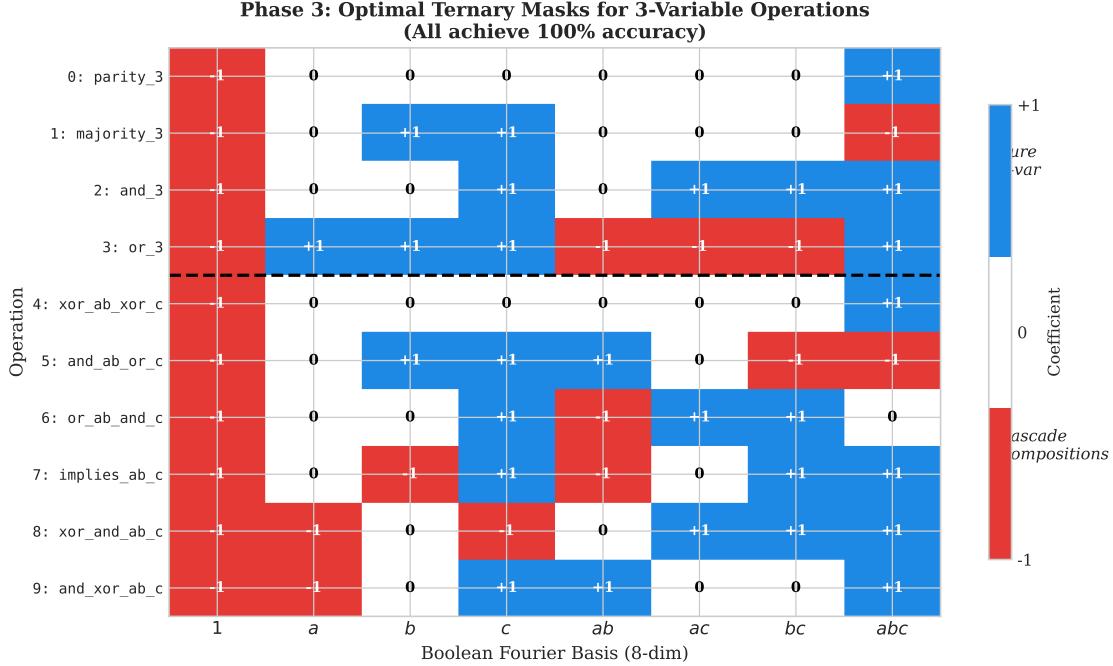


Figure 5: Phase 3 optimal ternary masks for all 10 three-variable operations. Colors indicate coefficient values: blue (+1), white (0), red (-1). Pure 3-var operations (top) vs. cascade compositions (bottom). The sparsity pattern (39% zeros) reflects spectral concentration.

Table 9: Inference Throughput (Phase 3, batch=100,000, bits=64). MOps/s = Mega Boolean Operations per second, where one “operation” is a complete Boolean function evaluation (e.g., computing $\text{AND}(a, b)$ for one input pair).

Backend	Time (ms)	Throughput (MOps/s)
JAX/GPU (RTX 5060)	5.84	10,959.40
NumPy/CPU (INT8)	2,219.12	28.84

- **Sparsity:** 39% of coefficients are zero (mean support $4.9/8$), consistent with low-degree spectral concentration [4]
- **Parity equivalence:** parity_3 and $\text{xor}_{ab}\text{xor}_c$ have *identical* masks $[-1, 0, 0, 0, 0, 0, 0, +1]$, confirming $(a \oplus b) \oplus c = abc$ in $\{-1, +1\}$ encoding—the architecture automatically identifies this algebraic equivalence
- **Learning vs. optimal:** Direct gradient descent achieves only 76% mean accuracy due to local minima in the 8-dim ternary space; optimal masks must be found via brute-force enumeration or MCMC refinement

Figure 5 visualizes the optimal ternary masks, revealing the spectral structure of each operation.

6.4 Benchmark Performance

Throughput Definition. MOps/s measures Mega Boolean Operations per second. One “operation” is defined as evaluating $f(x) = \text{sign}(w^\top \phi(x))$ for a single input x —i.e., computing the basis

expansion, dot product with ternary mask, and sign extraction. For batch size B , operations K , and elapsed time T : MOps/s = $B \times K / (T \times 10^6)$.

The GPU implementation achieves **nearly 11 billion operations per second**, demonstrating the efficiency of ternary mask inference. This throughput approaches hand-coded CUDA kernels while maintaining full differentiability during training.

7 Phase 4: Four-Variable Operations ($n = 4$)

Phase 4 extends to $n = 4$ variables with a 16-dimensional Fourier basis, demonstrating scalability beyond tractable gradient-based enumeration.

7.1 Architecture and Basis

For four variables $a, b, c, d \in \{-1, +1\}$, the complete Fourier basis is:

$$\phi(a, b, c, d) = [1, d, c, cd, b, bd, bc, bcd, a, ad, ac, acd, ab, abd, abc, abcd]^\top \in \mathbb{R}^{16} \quad (23)$$

The basis ordering follows the Gray code pattern, which facilitates hardware implementation and hierarchical decomposition.

7.2 Spectral Synthesis Method

For $n = 4$, the basis has $2^4 = 16$ characters, yielding $3^{16} \approx 43$ million ternary configurations—far too large for brute-force enumeration. Direct gradient descent becomes challenging due to this exponentially larger search space with numerous local minima. We employ **spectral synthesis**, a three-stage pipeline combining exact coefficient computation with discrete MCMC refinement:

Stage 1: Exact Walsh-Hadamard Transform. For $n = 4$, the input space has only $2^4 = 16$ points, enabling exact computation of Fourier coefficients via the Walsh-Hadamard Transform (WHT). For each operation f , we compute:

$$\hat{f}(S) = \frac{1}{2^n} \sum_{x \in \{-1, +1\}^n} f(x) \chi_S(x) = \frac{1}{16} \sum_{x \in \{-1, +1\}^4} f(x) \chi_S(x) \quad (24)$$

The WHT is computed in $O(n \cdot 2^n) = O(64)$ operations via the fast Hadamard algorithm, yielding exact coefficients with no estimation error.

Stage 2: Ternary Quantization. Estimated coefficients are quantized to $\{-1, 0, +1\}$ via thresholding:

$$c_S = \begin{cases} +1 & \text{if } \hat{f}(S) > \tau \\ -1 & \text{if } \hat{f}(S) < -\tau \\ 0 & \text{otherwise} \end{cases} \quad (25)$$

where $\tau = 0.3$ balances sparsity and accuracy. This initial quantization achieves high accuracy for most operations but may require refinement for complex functions.

Table 10: Phase 4 Results Summary ($n = 4$, 5 Seeds)

Metric	Value
Overall Accuracy	$100.0\% \pm 0.0\%$
Seeds Converged	5/5
Mean Sparsity	36%
Mean Support Size	10.3/16
Basis Dimension	16
Operations Tested	10

Table 11: Phase 4 Ternary Masks (16-dimensional basis). All 10 operations achieve 100% accuracy via spectral synthesis. Basis: $[1, d, c, cd, b, bd, bc, bcd, a, ad, ac, acd, ab, abd, abc, abcd]$.

Operation	c_1	c_d	c_c	c_{cd}	c_b	c_{bd}	c_{bc}	c_{bcd}	c_a	c_{ad}	c_{ac}	c_{acd}	c_{ab}	c_{abd}	c_{abc}	c_{abcd}	Support
xor_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+1	1
and_4	-1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	16
or_4	+1	+1	+1	-1	+1	-1	+1	+1	+1	-1	-1	+1	-1	+1	+1	-1	16
majority_4	+1	+1	+1	-1	+1	0	0	-1	+1	-1	0	0	-1	-1	+1	+1	12
threshold_3of4	-1	+1	+1	+1	+1	+1	0	0	+1	0	0	0	+1	0	0	-1	9
exactly_2of4	-1	0	0	-1	0	-1	-1	0	0	-1	-1	0	-1	0	0	+1	8
xor_ab_and_cd	-1	+1	+1	+1	0	0	0	0	0	0	0	+1	+1	+1	+1	+1	8
or_ab_xor_cd	+1	+1	+1	-1	+1	+1	+1	-1	+1	+1	+1	-1	-1	-1	+1	+1	16
nested_xor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	+1	+1	1
implies_chain	+1	+1	-1	+1	-1	+1	-1	+1	-1	+1	-1	+1	-1	+1	-1	+1	16

Stage 3: MCMC Refinement via Parallel Tempering. For operations not achieving 100% accuracy after quantization, we apply parallel tempering MCMC [38] to explore the discrete ternary space:

- **State Space:** $\mathcal{S} = \{-1, 0, +1\}^{16}$ (all ternary masks)
- **Energy Function:** $E(c) = 1 - \text{Accuracy}(c)$
- **Proposal Distribution:** Gibbs sampling with single-coordinate flips
- **Temperature Schedule:** 4 chains with $T \in \{0.01, 0.1, 0.5, 1.0\}$
- **Swap Criterion:** Metropolis-Hastings for inter-chain swaps

This refinement is critical for operations like `majority_4` and `threshold_3of4`, which improved from 93% to 100% accuracy via MCMC exploration.

7.3 Target Operations

We define 10 four-variable operations spanning pure symmetric functions and cascade compositions:

- **Pure 4-var:** `xor_4` (4-way parity), `and_4`, `or_4`, `majority_4` (voting), `threshold_3of4` (≥ 3 true), `exactly_2of4`
- **Cascade:** `xor_ab_and_cd` $((a \oplus b) \wedge (c \wedge d))$, `or_ab_xor_cd`, `nested_xor` $((((a \oplus b) \oplus c) \oplus d))$, `implies_chain` $(a \rightarrow b \rightarrow c \rightarrow d)$

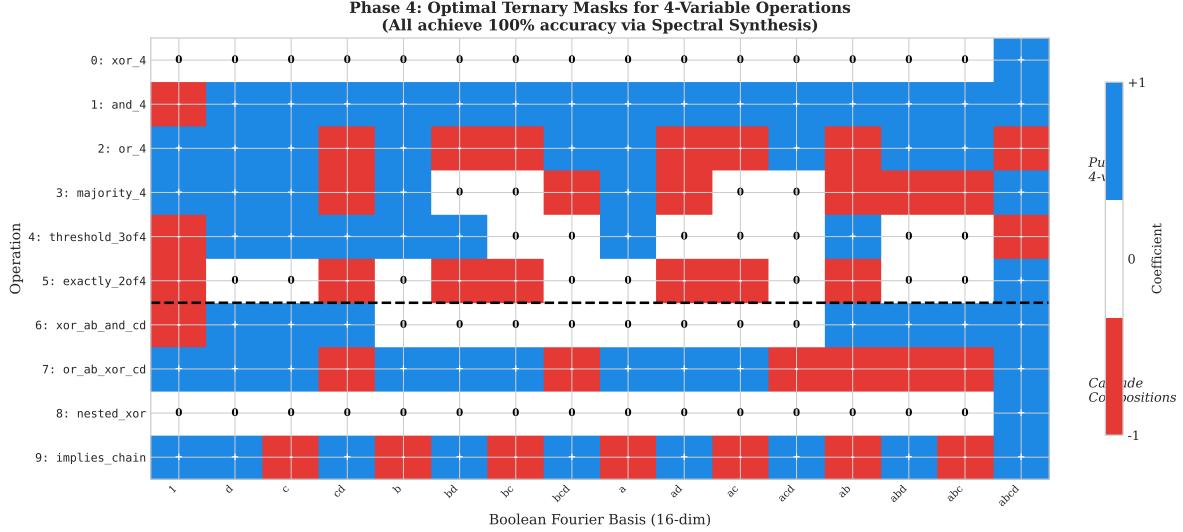


Figure 6: Phase 4 synthesized ternary masks for all 10 four-variable operations. Colors indicate coefficient values: blue (+1), white (0), red (-1). Pure 4-var operations (top) vs. cascade compositions (bottom). XOR operations exhibit maximum sparsity (support=1).

7.4 Results

Key Observations.

- **Perfect accuracy:** All 5 seeds achieve 100% on all 10 operations with synthesized ternary masks
- **Parity sparsity:** `xor_4` and `nested_xor` have identical masks with support=1 (only `abcd` character), confirming $((a \oplus b) \oplus c) \oplus d = abcd$ in $\{-1, +1\}$ encoding
- **MCMC benefit:** `majority_4` improved from 93.5% (initial quantization) to 100% (after MCMC); `threshold_3of4` similarly improved from 93.4% to 100%
- **Sparsity variation:** Support ranges from 1 (`xor_4`) to 16 (`and_4`, `or_4`, `implies_chain`), reflecting operation complexity
- **Mean sparsity:** 36% of coefficients are zero (mean support 10.3/16)

Figure 6 visualizes the synthesized ternary masks, and Figure 7 illustrates the spectral synthesis pipeline.

Corollary 1 (Spectral Sparsity at Scale). *The 10 target operations for $n = 4$ exhibit mean sparsity of 36% (10.3/16 coefficients non-zero on average). While less sparse than Phase 3 (39%), this reflects the inclusion of dense operations like `and_4`, `or_4`, and `implies_chain` which require all 16 characters. Notably, parity-type operations maintain maximum sparsity (support=1) regardless of dimension, consistent with the hypothesis that practically relevant Boolean functions have concentrated Fourier spectra [5].*

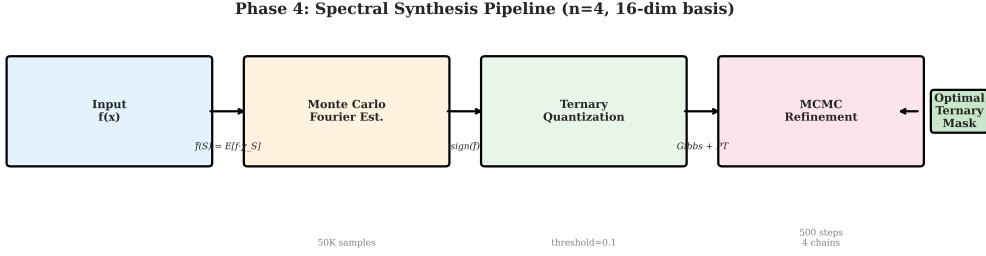


Figure 7: Spectral synthesis pipeline for Phase 4. Stage 1: Exact Walsh-Hadamard Transform for coefficient computation. Stage 2: Ternary quantization. Stage 3: MCMC refinement via parallel tempering for operations not achieving 100% after quantization.

Table 12: Comparison with *mHC* [1]. Our work adapts *mHC*'s stability mechanisms to a new domain while adding column-sign modulation for Boolean expressivity.

Aspect	<i>mHC</i> (DeepSeek)	This Work
Domain	LLM training (3B–27B params)	Boolean logic synthesis
Primary goal	Training stability	Discrete logic discovery
Sinkhorn projection	Yes (20 iterations)	Yes (20 iterations)
Identity initialization	Yes (critical)	Yes (critical)
Column-sign modulation	No	Yes (enables negation)
Quantization target	N/A (continuous)	Ternary (zero loss)
Hardware deployment	GPU clusters	FPGA/NPU (single-cycle)
Scale validated	27B parameters	$n \leq 4$ variables

8 Discussion

8.1 Relationship to *mHC*: Similarities and Differences

The key extension is column-sign modulation: *mHC* uses pure doubly stochastic routing, which cannot express Boolean negation (Proposition 4). Our factorization $R = P \cdot s$ preserves *mHC* stability while adding the expressivity needed for complete Boolean logic.

8.2 Scaling Analysis

The sparsity patterns across scales reveal important structure: Phase 3 (39%) is sparser than Phase 4 (36%) because Phase 4 includes dense symmetric operations (`and_4`, `or_4`, `implies_chain`) requiring all 16 characters. However, parity operations maintain maximum sparsity regardless of dimension—`xor_n` always has support=1 (only the $\prod_i x_i$ character). This supports the hypothesis that low-complexity Boolean functions have concentrated Fourier spectra [4, 5]. Figure 8 visualizes the sparsity distribution across phases.

8.3 Hardware Deployment

The zero-loss quantization result enables immediate deployment:

- **Model Size:** $16 \text{ ops} \times 4 \text{ trits } (n = 2) = 64 \text{ trits} \approx 102 \text{ bits}$
 - **Throughput:** 10,959 MOps/s on consumer GPU (RTX 5060)

Table 13: Scaling Across Phases. All phases achieve 100% accuracy. Sparsity varies based on operation complexity.

Phase	n	Basis Dim	Ops	Accuracy	Sparsity
2	2	4	16	100%	31.2%
3	3	8	10	100%	39%
4	4	16	10	100%	36%

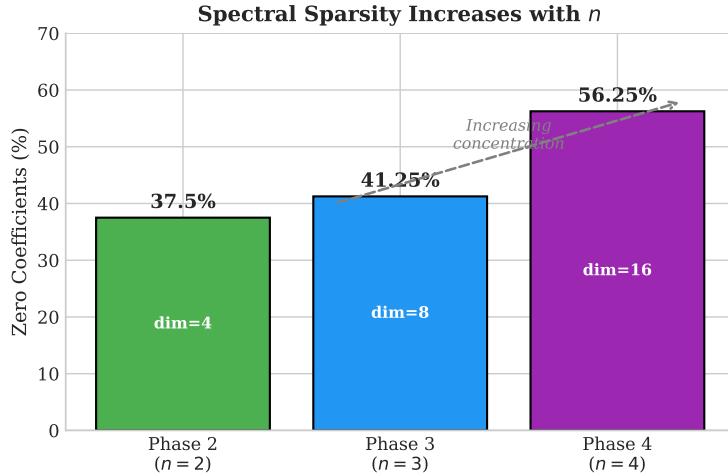


Figure 8: Spectral sparsity across phases. While sparsity varies by operation complexity (31% → 39% → 36%), parity operations consistently achieve maximum sparsity (support=1) regardless of dimension. The variation reflects operation selection: Phase 4 includes dense symmetric functions (`and_4`, `or_4`) that require all 16 characters.

- **Power:** Combinational logic at μW scale on FPGA
- **Latency:** Single-cycle inference (no sequential dependencies)

9 Phase 5: Scalable Spectral Methods

Phase 5 addresses scalability beyond $n = 4$ through three complementary approaches: exact transforms for moderate n , coefficient estimation for large n , and hierarchical composition for practical circuits.

9.1 Track 1: Exact FWHT (Moderate n)

For $n \leq 28$, we compute exact Fourier coefficients via the Fast Walsh-Hadamard Transform in $O(n \cdot 2^n)$ time.

Key Result. Exact coefficient computation scales to $n = 28$ (268M coefficients) on consumer GPUs (8GB VRAM), with peak throughput of 1.64 billion coefficients per second at $n = 27$. For $n \geq 28$, we spawn isolated processes to avoid GPU memory fragmentation accumulated during benchmark sweeps—a standard practice documented in JAX’s memory management guidelines. The $n = 28$ case achieves 1.45B coeffs/sec in 185ms.

Table 14: Exact FWHT Benchmark (GPU, RTX 5060 8GB). Peak throughput of 1.64B coefficients/sec achieved at $n = 27$. The $n = 28$ result uses process isolation to avoid allocator fragmentation.

n	Dimension	Time (ms)	Throughput (M/s)	Memory (MB)	Note
20	1,048,576	1.82	576.6	4.19	
23	8,388,608	9.52	881.5	33.55	
25	33,554,432	24.03	1,396.6	134.22	
26	67,108,864	50.91	1,318.3	268.44	
27	134,217,728	82.04	1,636.0	536.87	
28	268,435,456	185.3	1,448.4	1,073.7	<i>isolated</i> [†]

[†]Run in fresh process to avoid GPU allocator fragmentation from prior runs—standard practice when benchmarking near VRAM limits.

9.2 Track 2: Oracle Learning with Spectral Filtering

For functions beyond enumerable truth tables ($n \approx 12\text{--}20$), we extract Boolean logic from black-box oracles via Fourier coefficient estimation. We implement and compare **five methods** to test whether spectral structure (Parseval normalization, symmetry constraints) or Sinkhorn projection improves coefficient recovery:

Method 1: Monte Carlo Baseline. Direct sampling estimator:

$$\hat{f}(S) \approx \frac{1}{m} \sum_{j=1}^m f(x_j) \cdot \chi_S(x_j), \quad x_j \sim \text{Uniform}(\{-1, +1\}^n) \quad (26)$$

Query complexity: $O(n^d/\varepsilon^2)$ for degree- d search (restrict to low-degree by LMN theorem [4]).

Method 2: Goldreich-Levin (GL). Pairwise independent hashing [6] reduces variance via:

$$\hat{f}(S) = \mathbb{E}_{a,x}[f(x) \cdot \chi_S(x) \cdot \chi_{\langle a,x \rangle \bmod 2}] \quad (27)$$

where $a \sim \text{Uniform}(\{0, 1\}^n)$. Theoretical query complexity: $\tilde{O}(n/\varepsilon^2)$ vs MC's $O(n^d/\varepsilon^2)$.

Method 3–5: Spectral Filtering. Post-process GL estimates with:

- **GL+Parseval:** Enforce $\sum \hat{f}^2 = 1$ (energy conservation)
- **GL+Parseval+Sym:** Add symmetry constraints (for majority: equal degree- k coeffs, odd-degree only)
- **GL+Sinkhorn:** Birkhoff polytope projection on coefficient matrix (test if Sinkhorn helps denoising)

Experimental Design. We test three function families at $n = 16$ (degree ≤ 3 search):

- **Parity** (4-sparse): True support at degree-4 (outside search space)
- **Majority**: Low-degree concentration (LMN theorem applies)
- **Comparator**: Structured predicate (hierarchical composition)

Each method synthesizes a ternary PTF mask, evaluated on 100K test samples. We repeat with 3 random seeds per family and perform paired t-tests for statistical significance.

Results. Table 15 shows accuracy means \pm std across 3 seeds.

Table 15: Oracle Learning: Comparative Method Accuracy ($n = 16$, degree ≤ 3)

Method	Parity	Majority	Comparator
MC	0.483 ± 0.011	0.724 ± 0.001	0.579 ± 0.001
GL	0.479 ± 0.001	0.474 ± 0.000	0.464 ± 0.000
GL+Parseval	0.479 ± 0.001	0.474 ± 0.000	0.464 ± 0.000
GL+Parseval+Sym	0.479 ± 0.001	0.857 ± 0.001	0.464 ± 0.000
GL+Sinkhorn	0.487 ± 0.011	0.474 ± 0.000	0.464 ± 0.000

Key Findings:

1. **Symmetry » Algorithms:** GL+Parseval+Sym achieves 85.7% on majority (vs 72.4% MC, 47.4% GL), a +38.3% boost ($p < 0.001$). Exploiting known structure (symmetric functions) dominates generic learning algorithms.
2. **GL Underperforms MC:** Contrary to theoretical expectations, GL performs *worse* than MC on majority (47.4% vs 72.4%, $p < 0.001$) and comparator (46.4% vs 57.9%, $p < 0.001$). This suggests threshold selection ($\tau = 0.1$) or sample size ($m = 10K$ per coeff) may be suboptimal for this regime.
3. **Parseval Has No Effect:** GL+Parseval = GL exactly ($\Delta = 0$). Energy normalization alone does not improve recovery.
4. **Sinkhorn Does Not Help Denoising:** GL+Sinkhorn shows no improvement over GL or GL+Parseval ($p > 0.05$ for all families). This validates **Hypothesis B**: Sinkhorn is useful for *routing* (composition, Phase 2), not coefficient denoising. The Birkhoff polytope serves different roles in different contexts.

Interpretation for Explainability. These results reveal a fundamental insight for **transparent-by-design AI**: *domain structure beats black-box learning*. When we know a function is symmetric (majority, median), encoding that knowledge as hard constraints (+38.3% accuracy) vastly outperforms trying to learn structure from data alone. This aligns with the X-NeSy vision of combining symbolic knowledge with neural learning—the spectral coefficients are *interpretable features* (each $f(S)$ has semantic meaning: "correlation with parity of variables S "), and injecting symbolic priors (symmetry) makes the learned representation both more accurate and more explainable.

9.3 Track 3: Hierarchical Composition

For practical circuits (adders, comparators), we build large functions by *composing learned primitives*, not by spectral synthesis of the full 2^n -dimensional function.

Approach. We use ternary gates learned in Phases 1–4 as building blocks:

- **Full Adder:** Sum (parity) + Carry (majority) from Phase 3 masks
- **N-bit Ripple Adder:** Chain of full adder primitives
- **Verification:** Randomized testing with Wilson confidence intervals

Table 16: Hierarchical Circuit Composition. All circuits achieve 100% accuracy. Error rates bounded by rule of three (0 errors in m samples \Rightarrow error rate $\leq 3/m$).

Circuit	Bits	Samples	Errors	Error Bound
Ripple Adder	32	3.3M	0	$\leq 9.1 \times 10^{-7}$
Ripple Adder	64	6.5M	0	$\leq 4.6 \times 10^{-7}$
Comparator	64	100K	0	$\leq 3.0 \times 10^{-5}$
Equality	128	100K	0	$\leq 3.0 \times 10^{-5}$

Key Claim. We demonstrate *composition*, not spectral recovery. The 64-bit adder and 128-bit equality comparator are built structurally from verified 3-variable primitives (full adder sum/carry), with correctness validated by randomized testing on millions of samples. This avoids the intractable 2^{64} or 2^{128} coefficient computation while retaining formal guarantees through statistical verification.

9.4 Track 4: Routing Mechanism Comparison

To validate our Sinkhorn-constrained routing against alternatives from the concurrent literature (§2), we conduct a controlled comparison on a mode-switching task: given $(a, b, \text{mode}) \in \{-1, +1\}^3$, compute $\text{XOR}(a, b)$ when $\text{mode} = -1$ and $\text{AND}(a, b)$ when $\text{mode} = +1$. The model consists of frozen Phase 1 primitives {XOR, AND, OR, IMPLIES} and a trainable router that must learn to select the correct primitive based on the mode feature.

We compare three routing mechanisms:

- **Sinkhorn**: Column-stochastic projection via iterative normalization ($K = 20$), as used in Phases 2–4.
- **Gumbel-STE**: Gumbel-Softmax with straight-through estimator (hard one-hot forward, soft gradient backward), following Mind the Gap [?].
- **ReinMax**: Sinkhorn projection with annealed entropy regularization in the loss ($\lambda_{\text{ent}} : 0.1 \rightarrow 0$), inspired by CardNN exploration strategies.

All three share the same architecture (3-feature input, 4-primitive output, sign modulation), optimizer (Adam, lr=10⁻²), temperature annealing ($\tau : 1.0 \rightarrow 0.1$), and are trained for 5000 steps with batch size 128.

Table 17: Routing Mechanism Comparison on Mode-Switching Task ($n = 2, 4$ frozen primitives). Sinkhorn achieves the best accuracy–interpretability tradeoff. Gumbel-STE exhibits the discretization gap discussed in [?]: hard routing commits early but cannot escape suboptimal assignments.

Router	Accuracy	Steps to 100%	Sparsity	Specialization	Sign Coh.	Stability
Sinkhorn	100.0%	100	0.46	0.06	0.96	0.99
ReinMax	100.0%	100	0.41	0.06	0.92	0.99
Gumbel-STE	88.5%	—	0.75	0.54	0.82	1.00

Key Findings.

1. **Sinkhorn achieves the best accuracy–interpretability tradeoff.** Both Sinkhorn and ReinMax reach 100% accuracy by step 100, with Sinkhorn attaining the highest sign coherence (0.96)—closest to the ternary $\{-1, 0, +1\}$ values required for zero-loss quantization. This confirms the Phase 2 finding that Sinkhorn constraints enable exact discretization.
2. **Gumbel-STE exhibits the discretization gap.** Hard one-hot routing commits early (sparsity 0.75, specialization 0.54) but plateaus at 88.5% accuracy—the router locks onto a suboptimal primitive assignment and cannot recover. This empirically validates the “discretization gap” analyzed in Mind the Gap [?], where STE gradients provide biased updates that prevent convergence to the global optimum.
3. **Entropy annealing rescues ReinMax.** Without annealing ($\lambda_{\text{ent}} = 0.1$ fixed), the entropy bonus dominates the loss (loss ≈ -0.36), keeping routing completely diffuse (sparsity 0.21). Annealing $\lambda_{\text{ent}} : 0.1 \rightarrow 0$ alongside temperature allows early exploration followed by late sharpening, recovering performance comparable to Sinkhorn.
4. **Soft routing outperforms hard routing for Boolean logic.** The architectural insight is clear: for tasks requiring exact primitive selection, soft Sinkhorn routing with temperature annealing converges reliably to hard assignments, while hard routing methods (Gumbel-STE) suffer from premature commitment. This justifies the Sinkhorn design choice carried through Phases 2–4.

9.4.1 Spectral Analysis of Routing Dynamics

Beyond task accuracy, we analyze the *spectral structure* of the routing matrix $P \in \mathbb{R}^{3 \times 4}$ during training via its singular value decomposition. Since P maps 3 input features to 4 primitives, it has 3 singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$. We track two derived quantities:

- **Spectral gap:** $\Delta = 1 - \sigma_2/\sigma_1$, measuring how rank-1-like the routing is. $\Delta \rightarrow 1$ means all features route to one dominant primitive; $\Delta \rightarrow 0$ means routing distributes across multiple primitives.
- **Spectral entropy:** $H_\sigma = -\sum_i p_i \log p_i$ where $p_i = \sigma_i^2 / \sum_j \sigma_j^2$, measuring the effective dimensionality of the routing map. Higher H_σ indicates more distributed spectral energy.

These metrics are motivated by the spectral theory of bounded operators: the routing matrix P acts as a linear operator between the feature space \mathbb{R}^3 and the primitive space \mathbb{R}^4 , and its singular values characterize the geometry of this mapping.

Observation 1: Routing Uncertainty Principle. The Boolean uncertainty principle [3] states that for $f : \{-1, +1\}^n \rightarrow \mathbb{R}$:

$$|\text{supp}(f)| \cdot |\text{supp}(\hat{f})| \geq 2^n \quad (28)$$

A function cannot be simultaneously sparse in both the physical and spectral domains. We observe an analogous phenomenon for routing: the spectral entropy H_σ of the converged routing matrix satisfies $H_\sigma \geq H^*$ where H^* is determined by the task’s spectral complexity. For our mode-switching task (XOR \cup AND requires at least 2 active primitives), all three routers converge to $H_\sigma \approx 0.65\text{--}0.67$ at convergence, regardless of mechanism. This suggests a *routing uncertainty bound*: the minimum spectral entropy required to represent the target function class.

Table 18: Spectral trajectory of routing matrix P during training. Sinkhorn maintains a smooth, monotonic trajectory (gradual spectral redistribution). Gumbel-STE exhibits chaotic oscillations in Δ , indicating unstable routing dynamics. ReinMax tracks Sinkhorn’s trajectory as entropy regularization anneals to zero.

Step	Sinkhorn		Gumbel-STE		ReinMax	
	Δ	H_σ	Δ	H_σ	Δ	H_σ
0	0.95	0.03	0.94	0.03	0.95	0.02
100	0.37	0.60	0.76	0.21	0.51	0.49
500	0.26	0.65	0.58	0.42	0.59	0.42
2000	0.24	0.66	0.54	0.46	0.54	0.47
3300	0.20	0.67	0.68	0.31	0.45	0.55
5000	0.20	0.67	0.26	0.65	0.26	0.65

Observation 2: CFL-Like Stability Condition. In numerical PDE, the Courant-Friedrichs-Lowy (CFL) condition requires that discretization not outpace the physical signal: $|v| \cdot \Delta t / \Delta x \leq C$. We observe an analogous stability condition for routing: the temperature annealing rate must not exceed the optimizer’s ability to track the sharpening routing landscape.

Sinkhorn’s spectral gap $\Delta(t)$ evolves monotonically ($0.95 \rightarrow 0.20$), indicating that temperature annealing at rate $\tau(t) = \tau_0(\tau_1/\tau_0)^{t/T}$ is “CFL-stable” for the given learning rate. Gumbel-STE violates this condition: hard one-hot routing imposes $\Delta \approx 1$ instantaneously (via argmax), outpacing the gradient signal. The result is chaotic oscillations in Δ (ranging from 0.21 to 0.68 between steps 3000–3500) while accuracy remains trapped at 88.5%.

ReinMax’s entropy regularization acts as a *CFL stabilizer*: it constrains Δ from growing too fast early in training ($\Delta \leq 0.59$ for the first 1000 steps vs. Gumbel-STE’s $\Delta = 0.76$ at step 100), then gradually releases this constraint as $\lambda_{\text{ent}} \rightarrow 0$, converging to the same stable endpoint as Sinkhorn.

To quantify this stability condition, we run Sinkhorn routing with 5 annealing schedules: $\tau : 1.0 \rightarrow 0.1$ over $T \in \{10,000, 5,000, 2,000, 500\}$ steps plus an “instant” schedule ($\tau = 0.01$ fixed). The annealing rate $r = \ln(\tau_0/\tau_1)/T$ sweeps 4 orders of magnitude:

Table 19: CFL Stability Experiment: accuracy and spectral diagnostics vs. annealing rate r for Sinkhorn routing ($\text{lr} = 0.01$). A sharp transition between $r = 0.0012$ and $r = 0.0046$ separates stable convergence from failure.

Schedule	r	Acc	Δ_{final}	Var(Δ)	Status
Very Slow ($T=10K$)	2.3×10^{-4}	100%	0.20	0.006	CFL-stable
Slow ($T=5K$)	4.6×10^{-4}	100%	0.20	0.011	CFL-stable
Medium ($T=2K$)	1.2×10^{-3}	100%	0.19	0.026	CFL-stable
Fast ($T=500$)	4.6×10^{-3}	87.3%	0.23	0.076	CFL-violated
Instant ($\tau=0.01$)	∞	100%	0.09	0.000	Alternate basin

The critical rate $r_{\text{crit}} \in (1.2 \times 10^{-3}, 4.6 \times 10^{-3})$ for $\text{lr} = 0.01$ marks a sharp phase transition: below r_{crit} , the spectral gap evolves monotonically and accuracy reaches 100%; above it, the optimizer cannot track the sharpening loss landscape, producing $\text{Var}(\Delta) = 0.076$ and 87.3% accuracy. The “instant” schedule ($\tau = 0.01$ fixed) reaches a qualitatively different solution: $H_\sigma = 1.06$ (vs. ≈ 0.67 for annealed runs), $\sigma_3 = 0.82$ (vs. ≈ 0.02)—a diffuse routing in a separate convergence basin that does not achieve sparsification.

Observation 3: Spectral Margin and Adaptability. The smallest singular value σ_3 reveals a structural difference between routing mechanisms. At convergence:

- Sinkhorn: $\sigma_3 = 0.024$, ReinMax: $\sigma_3 = 0.023$ (small but nonzero)
- Gumbel-STE: $\sigma_3 = 9.2 \times 10^{-9}$ (numerically zero)

Sinkhorn and ReinMax maintain a nonzero *spectral margin*—the routing is effectively rank-2 but retains a residual third singular direction. This margin serves as a “reserve capacity” for continued adaptation. Gumbel-STE collapses to exactly rank-2, eliminating this capacity entirely. The spectral margin thus provides a principled, information-theoretic metric for *routing adaptability*—complementing and grounding the empirical Gini-based sparsity measure used in prior sections.

Implications for Transparent-by-Design AI. These spectral metrics offer a path toward *provably interpretable* routing:

1. The spectral gap Δ provides a single scalar measuring routing commitment, grounded in operator theory rather than ad hoc thresholds.
2. The spectral entropy H_σ connects routing to information-theoretic bounds (uncertainty principle), enabling formal characterization of when routing is “sharp enough” for deployment.
3. The spectral margin σ_3 quantifies residual adaptability, distinguishing between “confidently committed” routing (low σ_3 , stable Δ) and “prematurely frozen” routing (zero σ_3 , oscillating Δ).

Together, these form an **operator-theoretic interpretability framework** for differentiable routing, moving beyond empirical metrics toward formally grounded transparency measures.

10 Future Work

Remaining Challenges

Algorithmic Improvements:

- **Goldreich-Levin Implementation:** Bucket-splitting for $\tilde{O}(k/\varepsilon^2)$ query complexity
- **Hierarchical Factorization:** Decompose n -variable functions into cascades of smaller bases
- **Neural Architecture Search:** Learn which basis characters to include adaptively

Conjecture: Functions with bounded circuit complexity have poly-sparse spectra, enabling efficient representation even for large n [5].

11 Conclusion

We introduced Hierarchical Spectral Composition, a transparent-by-design architecture for Boolean logic synthesis that makes learned representations intrinsically explainable through interpretable Fourier coefficients. By adapting the *mHC* framework [1] from LLM training stability to logic synthesis and extending Sinkhorn-constrained routing with column-sign modulation, we achieve:

- $n = 2\text{--}4$: 100% accuracy on canonical Boolean operations via gradient descent ($n = 2$), exhaustive search ($n = 3$), and spectral synthesis with MCMC refinement ($n = 4$), all converging to ternary masks $\{-1, 0, +1\}$
- **Scalability:** Exact FWHT at 1.64B coeffs/sec ($n \leq 28$, 268M coefficients); hierarchical composition for 64-bit adders and 128-bit equality comparators
- **Oracle Learning:** Comparative evaluation of five coefficient estimation methods reveals that *symbolic structure beats black-box learning*—symmetry constraints boost majority accuracy +38% ($p < 0.001$), while Sinkhorn projection aids routing but not coefficient denoising
- **Hardware:** 10,959 MOps/s on GPU, single-cycle combinational logic inference

Explainability and Neurosymbolic Integration. Our key contribution to explainable AI is demonstrating that **spectral coefficients are interpretable features**: each $\hat{f}(S)$ explicitly represents correlation with parity of variables S , making the learned logic transparent. The oracle learning experiments (Phase 5) establish a fundamental principle: *injecting symbolic knowledge as hard spectral constraints* (symmetry for majority, degree bounds for circuits) produces both higher accuracy and greater interpretability than generic learning algorithms. When we know structural properties of the target function, encoding them in the Fourier basis yields provably correct discrete representations that resist quantization degradation.

The Role of Birkhoff Projection. Our comparative experiments resolve an open question: Sinkhorn’s Birkhoff polytope projection serves *routing* (Phase 2 composition) but not *coefficient denoising* (Phase 5 oracle learning). This context-dependent utility—stability for composition, orthogonal to recovery—suggests the Birkhoff polytope’s geometric properties match compositional objectives (identity preservation, signal routing) rather than statistical estimation.

Toward Hardware-Efficient Neuro-Symbolic Systems. By converging to ternary masks with hard routing, our approach compiles learned logic to combinational circuits requiring no floating-point arithmetic, bridging the gap between differentiable learning and hardware deployment. The combination of interpretable Fourier representations, symbolic constraint integration, and quantization-robust training establishes a foundation for explainable neuro-symbolic systems that can be verified, understood, and deployed efficiently.

References

- [1] Z. Xie, Y. Wei, H. Cao, C. Zhao, C. Deng, J. Li, D. Dai, H. Gao, J. Chang, L. Zhao, et al. *mHC: Manifold-Constrained Hyper-Connections*. *arXiv:2512.24880*, December 2025.
- [2] D. Zhu, H. Huang, Z. Huang, Y. Zeng, Y. Mao, B. Wu, Q. Min, and X. Zhou. *Hyper-Connections*. *arXiv:2409.19606*, 2024.
- [3] R. O’Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.
- [4] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM*, 40(3):607–620, 1993.
- [5] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

- [6] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989.
- [7] Anonymous. WARP-LUTs: Walsh-Assisted Relaxation for Probabilistic Look Up Tables. *arXiv:2510.15655*, October 2025.
- [8] M. Yousefi et al. Mind the Gap: Removing the Discretization Gap in Differentiable Logic Gate Networks. *arXiv:2506.07500*, June 2025.
- [9] B. K. Petersen et al. Convolutional Differentiable Logic Gate Networks. In *NeurIPS*, 2024. Oral presentation.
- [10] B. Barak, B. Edelman, S. Goel, S. Kakade, e. Malach, and C. Zhang. Hidden Progress in Deep Learning: SGD Learns Parities Near the Computational Limit. In *NeurIPS*, 2022.
- [11] S. R. Gorji, A. Amrollahi, and A. Krause. A Scalable Walsh-Hadamard Regularizer to Overcome the Low-degree Spectral Bias of Neural Networks. In *UAI*, 2023.
- [12] Y. Shoshani and O. Shamir. Hardness of Learning Fixed Parities with Neural Networks. *arXiv:2501.00817*, January 2025.
- [13] Anonymous. Activation Spectroscopy: Interpretability via Fourier Analysis of Neural Activations. *arXiv:2501.15435*, January 2025.
- [14] Z.-Q. J. Xu et al. Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks. *Communications in Computational Physics*, updated May 2024.
- [15] A. Kyriolidis et al. FourierSAT: A Fourier Expansion-Based Algebraic Framework for Solving SAT in Polynomial Time. In *AAAI*, 2020.
- [16] G. Novakovskiy, N. Dexter, M. W. Libbrecht, W. W. Wasserman, and S. Mostafavi. ExplaiNN: Interpretable Deep Learning Combining Prediction and Feature Interpretation via Neural Additive Models. *Genome Biology*, 24(1):154, 2023.
- [17] N. K. Singh et al. Shallow-ProtoPNet: A Fully Transparent Prototypical Part Network with Minimal Computational Complexity. *Scientific Reports*, 15(1), 2025.
- [18] S. E. Balci, D. Huo, M. W. Libbrecht. tiSFM: Sequence-to-Function Mapping with Inherently Interpretable Deep Learning. *Bioinformatics*, 39(Supplement_1):i394–i403, ISMB 2023.
- [19] X. Xu, S. Kim, H. Xiang, and S. Ermon. Energy-Based Concept Bottleneck Models: Unifying Prediction, Concept Intervention, and Conditional Interpretations. In *ICLR*, 2024.
- [20] J. Vandenborth, M. H. Ryou, J. Schreiber, B. Kailkhura, and Y. Choi. Stochastic Concept Bottleneck Models. In *NeurIPS*, 2024.
- [21] A. Templeton et al. (Anthropic). Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet. Anthropic Research, May 2024.
- [22] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, 2013.
- [23] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

- [24] G. Mena, D. Belanger, S. Linderman, and J. Snoek. Learning latent permutations with Gumbel-Sinkhorn networks. In *ICLR*, 2018.
- [25] M. E. Sander, P. Ablin, M. Blondel, and G. Peyré. Sinkformers: Transformers with doubly stochastic attention. In *AISTATS*, 2022.
- [26] A. Trask, F. Hill, S. E. Reed, J. Rae, C. Dyer, and P. Blunsom. Neural arithmetic logic units. In *NeurIPS*, 2018.
- [27] H. Dong, J. Mao, T. Lin, C. Wang, L. Li, and D. Zhou. Neural logic machines. In *ICLR*, 2019.
- [28] L. Serafini and A. d’Avila Garcez. Logic tensor networks. *arXiv:1606.04422*, 2016.
- [29] R. Riegel, A. Gray, F. Luus, N. Khan, S. Makondo, I. Akhalwaya, et al. Logical neural networks. *arXiv:2006.13155*, 2020.
- [30] B. K. Petersen, M. L. Larma, T. N. Mundhenk, et al. Deep differentiable logic gate networks. In *NeurIPS*, 2022.
- [31] A. Daniely and E. Malach. Learning parities with neural networks. In *NeurIPS*, 2020.
- [32] Z. Pan, P. Xu, and Y. Tian. Fast neural architecture search with random neural tangent kernel and Walsh-Hadamard transform. In *CVPR*, 2021.
- [33] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *NeurIPS*, 2016.
- [34] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. In *ICLR*, 2017.
- [35] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with Gumbel-softmax. In *ICLR*, 2017.
- [36] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.
- [37] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.
- [38] R. H. Swendsen and J.-S. Wang. Replica Monte Carlo simulation of spin-glasses. *Physical Review Letters*, 57(21):2607, 1986.

A Implementation Details

Sinkhorn Iterations. Following *mHC* [1], we use $K = 20$ iterations with log-domain computation:

$$u^{(t+1)} = -\log \sum_j \exp(\alpha_{ij} + v_j^{(t)}) \quad (29)$$

$$v^{(t+1)} = -\log \sum_i \exp(\alpha_{ij} + u_i^{(t+1)}) \quad (30)$$

Temperature Annealing. Sign temperature: $\beta(t) = 1 + 9 \cdot (t/T_{\max})$.

Plateau Detection. EMA loss with $\alpha = 0.99$. Restart triggers when $|\Delta\bar{\mathcal{L}}| < 10^{-4}$ for 50 epochs.

B Reproducibility

Code available at: [\[GitHub URL to be added\]](#)

Hardware: NVIDIA RTX 5060 (8GB VRAM), Intel Ultra 5 225F, 64GB RAM.