LOGICAL NEURAL UNIT: ORDER-STATISTIC NEURON

Anonymous authors
Paper under double-blind review

ABSTRACT

The core primitives of modern networks (inner products, attention, and gating) all aggregate by weighted sums, and a weighted sum responds to every input at once. So they are a poor fit for *sparse logical structure*: a target whose label is set by a few of many features through AND/OR/threshold logic. On such targets the relevant signal is diluted by the irrelevant ones, and the samples needed to recover it grow with the feature count, the regime where decision trees and rule learners still outperform deep networks on scientific data. But logic is not built from sums: AND and OR are extrema (min and max), each decided by a single coordinate, and with negation $1-x$ they form the complete basis for such targets. No dominant neural primitive computes an extremum; the networks that do implement logic directly (logic-gate and lookup-table nets) require binarized inputs and target hardware cost, and tree and rule learners are not differentiable. We introduce **Logical Neural Units (LNUs)**: differentiable order-statistic neurons that compute soft AND (softmin) and OR (softmax) over $[x, 1-x]$, with a sharpness knob from averaging to exact logic. Their gradient concentrates on a single coordinate with sensitivity independent of D; we prove a PAC sample-complexity separation: LNU's sample requirement grows only logarithmically with the feature count, versus linearly for MLPs, Transformers, and prior differentiable-logic neurons ($O(k\ell \log D)$ versus $\Omega(D)$). Empirically, under one matched protocol, LNU reaches 92–100% on non-parity canonical Boolean formulas (neural baselines below 79% at D=200), loses 0.6pp under 16:1 distractors (versus 21.8pp for an MLP), beats DWN by 4–10pp on continuous logical tasks, and outperforms tree ensembles and MLPs on real-world few-shot scientific data (DNA splice sites: +4pp over XGBoost, +8pp over MLP). LNU is a drop-in structural inductive bias, useful when the target is Boolean-structured; we report where it is not.

1 INTRODUCTION

Modern networks aggregate by weighted sums; logic does not. The dominant neural primitives, the **inner product** (weighted sum), **pairwise attention** (query-key softmax), and **multiplicative gating** (SiLU/GELU gates in modern FFN), all aggregate by weighted sums, and a weighted sum is sensitive to every coordinate at once: under random binary inputs $\sum w_i x_i$ has variance $\Theta(D)$. So they are mismatched to **sparse logical structure**: a target that is a Boolean function (a rule of ANDs, ORs, and NOTs) of a few relevant features hidden among many irrelevant ones, learned from few labeled examples (e.g. a splice site set by a few nucleotides, or a device fault by a few sensor thresholds). This is the structure that keeps decision trees and rule learners competitive on tabular and scientific data, where deep networks still underperform tree ensembles (Grinsztajn et al., 2022); yet on such targets a weighted sum dilutes the relevant signal at $O(1/\sqrt{D})$ (Proposition 2) and identifying which features matter costs $\Omega(D)$ samples (Theorem 4). Sparse structure, high dimension, low sample count: this is where standard neural networks struggle most.

Boolean logic points to the missing ingredient: AND and OR are decided not by a sum but by *extrema*. The Gödel t-norm (Hájek, 1998) sets $\mathrm{AND}(x) = \min_i x_i$ and $\mathrm{OR}(x) = \max_i x_i$, each determined by a *single* coordinate, and with negation $\mathrm{NOT}(x) = 1-x$ they form a functionally complete Boolean basis. Yet none of the dominant neural primitives computes a min or a max. The methods that *do* compute logic directly, logic-gate and lookup-table networks (Petersen et al., 2022; Bacellar et al., 2024), require binarized inputs and target hardware cost rather than sample-efficient

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

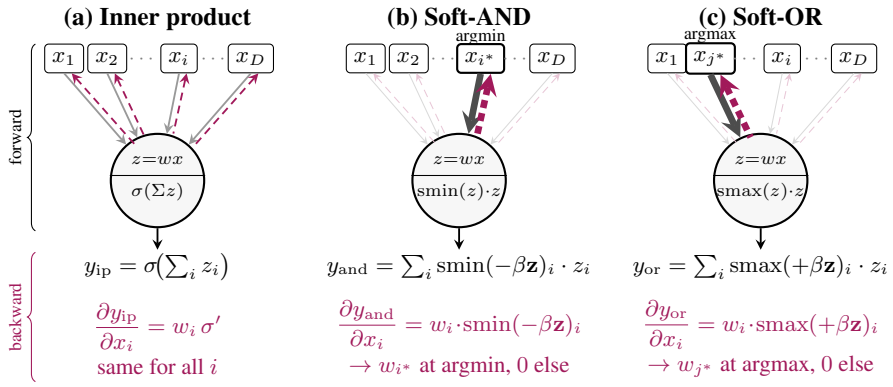


Figure 1: **Three neurons sharing $z_i{=}w_i x_i$ (upper half), differing in aggregation (lower half).** Solid = forward, dashed = gradient; thickness = magnitude. **(a)** Inner product: $\sigma(\sum z_i)$, gradient uniform. **(b)** Soft-AND: $\mathrm{smin}(z){\cdot}z$ with $\mathrm{smin}_i{=}\mathrm{softmax}(-\beta z)_i$; gradient on argmin. **(c)** Soft-OR: $\mathrm{smax}(z){\cdot}z$ with $\mathrm{smax}_i{=}\mathrm{softmax}(+\beta z)_i$; gradient on argmax. Adding features does not change argmin/argmax $\Rightarrow$ (b),(c) are D-invariant; (a) is not.

learning over continuous features, and tree and rule learners are not differentiable. No primitive computes such logic, learns it sample-efficiently, takes continuous inputs, and drops into a standard network.

This paper introduces Logical Neural Units (LNUs) (Figure 1): differentiable order-statistic neurons that apply softmin (AND) and softmax (OR) over the complement-augmented input $[x, 1{-}x]$, with a sharpness knob β from averaging to exact logic. Our central claim is that LNU learns sparse logical targets from far fewer samples than inner-product models, and we prove why (Theorem 4). As convolutional priors exploit translation equivariance and attention exploits pairwise similarity, LNU exploits Boolean-target structure, a structural prior for learning from sparse high-D data.

Contributions. Order-statistic operators (Yager, 1988; Martins & Astudillo, 2016) and differentiable soft-logic neurons (Payani & Fekri, 2019; Riegel et al., 2020) both exist; LNU's claim is not the min/max form but a property none has, a per-coordinate gradient independent of D and the sample-complexity separation it yields for logical targets. **(1) A missing primitive.** No standard neural aggregator is at once differentiable, continuous-input, sample-efficient, and drop-in for Boolean-structured targets: sum-based primitives (inner product, attention, gating) dilute sparse signal, logic-gate and LUT nets (Petersen et al., 2022; Bacellar et al., 2024) need binarized inputs and target hardware, and tree and rule learners are not differentiable. LNU fills this gap. **(2) Why it works.** LNU's per-coordinate gradient is $\Theta(|w_{i^*}|)$, independent of D (Proposition 3), where inner products dilute at $O(1/\sqrt{D})$ and prior soft-logic aggregators (Payani & Fekri, 2019; Riegel et al., 2020) collapse (Propositions 1, 2); this places LNU in a sparse hypothesis class with a PAC separation, $O(k\ell \log D)$ vs $\Omega(D)$ (Theorem 4). The separation is statistical, not representational (the witness is linearly separable), and is the formal reason for the measured sample-efficiency. **(3) A practical advantage, under one matched protocol.** Same seeds, splits, and budget across methods: LNU loses 0.6pp under 16:1 distractors (MLP -21.8pp), beats DWN by 4–10pp on continuous logical tasks because DWN must binarize, wins 1–8pp on four real-world few-shot datasets, and drops into OLMoE-1B-7B at parity with 31% fewer per-expert parameters. Scope boundaries (XOR, bAbI (Weston et al., 2015), semantic NLP) are reported in the appendix, not hidden.

Scope. LNU is a structural inductive bias, useful exactly when the target has the structure it assumes, as convolution helps on images or attention on sequences. That structure is a **sparse Boolean function**; the inputs themselves may be binary *or* continuous (on Gaussian features LNU still leads by up to $+22$pp, §5.3). When the assumption fails (large N, or dense targets *without* logical structure), inner-product models catch up or overtake, which we report (Tox21, native 20 Newsgroups, full-data FOLIO) rather than hide.

2 Background and Related Work

The inner-product barrier. An MLP neuron computes $h_j = \sigma(\sum_i w_{ji}x_i + b_j)$: a weighted sum followed by a pointwise nonlinearity. On Boolean targets over D binary features, three decades of theory establish that weighted sums face structural barriers: the VC-dimension of all Boolean functions is 2^D (Vapnik & Chervonenkis, 1971), UAT guarantees approximation on compact sets but not correct Boolean inference (Hornik et al., 1989; Kearns & Vazirani, 1994), and bounded-depth circuits require super-polynomial size for PARITY (Håstad, 1986). The practical consequence: as D grows, a single inner-product layer dilutes each coordinate's relative influence at rate $O(1/\sqrt{D})$ (Proposition 2), and MLPs plateau at the majority-class prior on AND/OR/MAJ targets at $D \geq 200$ (§5.1).

Prior soft-Boolean aggregators. Two differentiable approximations of AND exist in the literature. **NLN** (Payani & Fekri, 2019): $\mathrm{AND}(x) = \prod_i (1 - w_i(1 - x_i))$, which decays exponentially in D (Proposition 1). **LNN** (Riegel et al., 2020): $\mathrm{AND}(x) = \max(0, \sum w_i x_i - T)$, which loses per-coordinate sensitivity at rate $O(1/\sqrt{D})$ by CLT concentration (Proposition 2). Both have gradients $\partial f / \partial x_i \to 0$ as $D \to \infty$. To our knowledge, LNU's softmin/softmax is the first *learnable* D-dimensional aggregator with D-independent sensitivity (Proposition 3). Max-pooling also has D-independent sensitivity but applies a fixed operation without learned feature selection: it cannot weight or negate individual coordinates. Additional related work (LTN, dILP, NLM, fuzzy logic, OWA operators) is in Appendix I.

Trainable Boolean and LUT primitives. A separate line makes look-up tables or Boolean functions directly trainable. Differentiable logic gate networks (DLGN) (Petersen et al., 2022) learn deep circuits of *two-input* gates via softmax over 16 gate types; the LUT family (DWN (Bacellar et al., 2024), LUTNet (Wang et al., 2019), LogicNets (Umuroglu et al., 2020), PolyLUT (Andronic & Constantinides, 2023), NeuraLUT (Andronic & Constantinides, 2024), Truth Table Net (Benamira et al., 2024), FINN (Umuroglu et al., 2017)) trains lookup-table contents, primarily for efficient FPGA inference. These share two properties that distinguish them from LNU: their inputs are binarized or low-bit quantized, and their logical unit is a small-fan-in gate (a two-input gate, or a k-input LUT, typically $k \leq 6$) composed by wiring, not a D-dimensional aggregator. Their objective is a hardware cost budget (LUT/area/latency/energy). LNU instead accepts continuous inputs, computes a differentiable D-dimensional order statistic, and targets sample-efficient learning; it is a statistical inductive bias, not a hardware-mapping target (extended discussion in Appendix I).

Method comparison. Appendix I (Table 8) compares LNU with prior primitives on five desiderata for a differentiable Boolean primitive (D-independent sensitivity, smooth-to-crisp control via β, functional completeness, formula extractability, and drop-in compatibility); LNU is the only one to satisfy all five.

3 Logical Neural Units

3.1 The LNU primitive

Given input $x \in [0,1]^D$ (or $\{0,1\}^D$), an LNU layer with H hidden units computes H soft-AND and H soft-OR features. Each soft-AND unit j forms weighted products, applies softmin to select the smallest, and outputs their weighted sum:

$$z_{ji} = x_i \cdot w_{ji}, \quad \alpha_{ji} = \frac{e^{-\beta\, z_{ji}}}{\sum_k e^{-\beta\, z_{jk}}}, \quad \mathrm{AND}_j = \textstyle\sum_i \alpha_{ji}\, z_{ji} \;\approx\; \min_i\, z_{ji} \tag{1}$$

Here $w_{ji} \in \mathbb{R}$ are unconstrained learned weights (negative values enable literal negation, e.g. $w = -1$), $\beta > 0$ is the sharpness parameter, and AND_j denotes the j-th unit's scalar output (not the Boolean AND function). Soft-OR uses a separate weight matrix and $+\beta$ (softmax instead of softmin):

$$\gamma_{ji} = \frac{e^{+\beta\, z_{ji}}}{\sum_k e^{+\beta\, z_{jk}}}, \quad \mathrm{OR}_j = \textstyle\sum_i \gamma_{ji}\, z_{ji} \;\approx\; \max_i\, z_{ji} \tag{2}$$

Complement augmentation. The equations above operate on x directly, which represents positive literals (x_i) but not negated literals ($\neg x_i$). To achieve functional completeness, we apply **complement augmentation**: $\bar{x} := \mathrm{Aug}(x) = [x, 1-x] \in [0,1]^{2D}$, replacing x with $\bar{x}$ in Eqs. (1)–(2) (so the sums run over $i = 1, \ldots, 2D$). This provides access to $\mathrm{NOT}(x_i) = 1-x_i$ through the second half of $\bar{x}$, enabling the representation of any Boolean function. All statements below are on this augmented input.

How this implements Boolean AND and OR. The equations apply to any $\bar{x} \in [0,1]^{2D}$; we illustrate with the binary case $x \in \{0,1\}^D$. Consider a single AND unit realizing $x_1 \wedge x_2$. Since softmin selects the *smallest* product, a conjunction should send any *failed* literal to the lowest value; this is what the complement coordinate achieves. The unit puts $w = -1$ on the complement entries $1-x_1$ and $1-x_2$ (and $w = 0$ elsewhere), so their products are $z = -(1-x_i) = x_i - 1 \in \{-1, 0\}$: 0 when the literal holds ($x_i = 1$) and -1 when it fails. Every other coordinate, relevant or distractor, has product 0. Hence a satisfied literal and a distractor both sit at 0, and only a failure dips to -1, so $\min_i z_i = 0$ exactly when both literals hold and -1 otherwise, i.e. $\mathrm{AND}_j = \mathrm{AND}(x_1, x_2) - 1$ (the constant is absorbed by the FC head). Because failures alone break the 0 floor, the minimum does not depend on how many distractors are present, giving the distractor invariance of Proposition 3. The weights are learned, not fixed: SGD converges to this selection (100% on AND, Table 1). **OR** is dual: it uses the positive coordinates with $w = +1$ on x_1, x_2, so $\max_i z_i = 1$ when any $x_k = 1$ (disjunction satisfied) and 0 otherwise, a satisfied literal rising to 1 above the shared 0 floor.

The key difference from inner product: a weighted sum $w^\top x$ averages all D coordinates, diluting each relevant feature's influence at rate $O(1/\sqrt{D})$ (Proposition 2). Softmin concentrates on the single smallest product, making the output *independent of the number of irrelevant features* (Proposition 3).

Sharpness knob β. At β=0, softmin/softmax reduce to uniform averaging (since $e^{-\beta z_i} \to 1$ uniformly). At $\beta \to \infty$, they converge to exact $\min/\max$ — the Gödel t-norm (Hájek, 1998); Figure 5 in the appendix. Higher β sharpens selection but causes gradient vanishing; a straight-through estimator (STE; Bengio et al. (2013)) with β_{back}=1 restores trainability at any β (Appendix J.11, Figure 8). We use β=10 as default (§5).

3.2 Architecture: Aug(1L)+FC

Each LNU unit computes a single AND or OR over the input independently; a **composition layer** is needed to combine them into a prediction. We use a linear FC head: $y = \mathrm{FC}([\mathrm{AND}_1, \ldots, \mathrm{AND}_H, \mathrm{OR}_1, \ldots, \mathrm{OR}_H])$. A linear combination of AND outputs implements disjunction (OR of ANDs = DNF), which suffices for all formulas tested. We call this the **Aug(1L)+FC** architecture.

Functional completeness. Aug(1L)+FC represents any k-term DNF with ℓ literals per term at $\beta \to \infty$ with $O(k\ell)$ nonzero weights, provided the FC head has at least two ReLU layers (Observation 1; a single linear FC suffices only for monotone formulas). Both AND and OR units are included for compositionality: DNF targets benefit from explicit disjunction (ablation in Appendix E). The trained models in our experiments use the linear FC head; the two-layer ReLU head is invoked only to prove exact representation of arbitrary non-monotone DNF. Both operate over the same soft-AND/OR units, so the analyzed and the implemented model coincide at the unit level (the D-independent gradient of Proposition 3) and differ only in the read-out.

Parameter count. LNU layer: $4DH$ parameters ($2D \times H$ for AND weights + $2D \times H$ for OR weights). FC head: $2H+1$. Total: $4DH + 2H + 1$. Augmentation doubles the input but is mathematically redundant for inner-product layers, so comparisons are parameter-fair. For continuous inputs, a preprocessing step maps values into LNU's operating range: sigmoid for raw continuous features, LayerNorm for pretrained transformer hidden states (§5.4). The core softmin/softmax mechanism is identical in all cases. Details in Appendix D; the sigmoid-preprocessing MLP baseline and the complement-augmentation ablation are in Appendices G and F.

4 THEORETICAL RESULTS

This section shows LNU's empirical advantage is *structural*: a gradient-level mechanism prior aggregators lack (§4.1), formalized as a PAC sample-complexity separation (§4.2).

4.1 WHY PRIOR AGGREGATORS FAIL AND LNU SUCCEEDS

Per-coordinate gradient sensitivity that does not vanish with D is the key requirement for learning sparse Boolean targets. We show that two prior differentiable AND approximations lack this property, while LNU's extremal aggregation provides it. Soft-AND and soft-OR are exact duals ($\mathrm{softmax}_\beta(z) = -\mathrm{softmin}_\beta(-z)$); we state the AND versions only.

Proposition 1 (NLN product underflow)**.** *The Neural Logic Network (Payani & Fekri, 2019) computes* $\mathrm{AND}_{\mathrm{NLN}}(x) = \prod_i (1 - w_i(1 - x_i))$. *When* $w_i(1 - x_i) \geq \tau$ *for at least a fraction* $\rho \in (0, 1)$ *of coordinates (ρD total), the output satisfies* $\mathrm{AND}_{\mathrm{NLN}} \leq (1 - \tau)^{\rho D}$. *Both output and gradient* ***decay exponentially in*** D.

Intuition. A product of D terms, each ≤ 1, shrinks exponentially as D grows, even if only a constant fraction of factors are bounded away from 1.

Proposition 2 (LNN threshold concentration)**.** *The Logical Neural Network (Riegel et al., 2020) computes* $\mathrm{AND}_{\mathrm{LNN}}(x) = \max(0, \sum w_i x_i - T)$ *with* $c \leq |w_i| \leq C$. *The per-coordinate influence, measured as the ratio of single-coordinate sensitivity to total standard deviation,* $|\partial \mathrm{AND}_{\mathrm{LNN}} / \partial x_j| / \sqrt{\mathrm{Var}(\sum w_i x_i)}$, *is* $O(1/\sqrt{D})$.

Intuition. A weighted sum of D random variables concentrates around its mean with standard deviation $\Theta(\sqrt{D})$. Each coordinate's contribution w_j is $O(1)$, so its influence relative to the total fluctuation is $O(1/\sqrt{D})$; it becomes undetectable as D grows.

Proposition 3 (LNU D-independent sensitivity)**.** *For a single AND unit (dropping the unit index j from §3), LNU computes* $\mathrm{AND}_{\mathrm{LNU}}(x) = \sum_i \mathrm{softmin}(\beta z)_i \cdot z_i$ *where* $z_i = \mathrm{Aug}(x)_i \cdot w_i$ *are the weighted products over* $n = 2D$ *augmented inputs. Let* $z_{(1)} \leq z_{(2)} \leq \ldots$ *be the order statistics,* r *the multiplicity of the minimum,* $g(x) := z_{(r+1)} - z_{(1)}$ *the gap,* $M := \max_i |z_i|$, *and* $g_0 := \inf_x g(x) > 0$ *(positive gap assumed). Then:*

1. *(Approximation)* $|\mathrm{AND}_{\mathrm{LNU}}(x) - z_{(1)}| \leq \frac{2M(n-r)}{r} e^{-\beta g(x)}$.
2. *(Sensitivity)* $\partial \mathrm{AND}_{\mathrm{LNU}} / \partial x_{i^*} = w_{i^*}(1/r + O(n\, e^{-\beta g_0})) = \Theta(|w_{i^*}|)$, ***independent of*** D.

Intuition. The error term contains $n{=}2D$, so the bound needs $\beta = \Omega(\log D / g_0)$; at $\beta{=}10, g_0{=}1$ this covers $D \leq e^{10} \approx 22{,}000$, beyond our range. The softmin weights concentrate on the minimum product (others get weight $\leq e^{-\beta g}$, however many), so the gradient flows through the argmin coordinate with sensitivity $\Theta(|w_{i^*}|)$, independent of D.

Proof. In Appendix A (the argmin coordinate has $\partial \mathrm{AND} / \partial x_{i^*} = w_{i^*}/r$, D-independent; non-argmin coordinates are suppressed by $\alpha_i \leq e^{-\beta g}$).

Summary. As D grows, NLN outputs collapse (Prop. 1) and LNN gradients vanish into noise (Prop. 2), while LNU keeps constant sensitivity (Prop. 3); §5 confirms this.

4.2 PAC SAMPLE-COMPLEXITY SEPARATION

The D-independent gradient of Proposition 3 has a formal consequence: it enables a sparse hypothesis class with low VC-dimension. Two supporting results (statements and proofs in Appendix A) establish the sparsity: Aug(1L)+FC exactly represents any k-term DNF with ℓ literals per term at $\beta \to \infty$ (Observation 1), using at most $\|W^{\mathrm{AND}}\|_0 \leq k\ell$ nonzero AND-layer weights, **independent of** D (Observation 5); this sparsity is what enters the VC analysis below. Hence:

Theorem 4 (PAC separation)**.** *Let* $\mathcal{F}_{k,\ell,D}$ *be the class of k-term DNF Boolean functions over* $\{0,1\}^D$ *with at most ℓ literals per term and relevant set* $|S| \leq k\ell$, *and let the input distribution be uniform on* $\{0,1\}^D$. *For any* $\varepsilon, \delta \in (0, 1/2)$*:*

- *(Upper bound) The class $\mathcal{H}^s_{\mathrm{LNU}}$ of Aug(1L)+FC networks with $\|W^{\mathrm{AND}}\|_0 \le s = k\ell$ PAC-learns $\mathcal{F}_{k,\ell,D}$ with $N_{\mathrm{LNU}} = O\big((k\ell\log(D/(k\ell)) + (k\ell)^2 + \log(1/\delta))/\varepsilon^2\big)$; for sparse targets with $k\ell = O(\log D)$ the $(k\ell)^2$ term is absorbed.*
- *(Lower bound) Any proper algorithm from $\mathcal{H}_{\mathrm{IP}}$ (linear threshold functions) requires $N_{\mathrm{IP}} = \Omega((D + \log(1/\delta))/\varepsilon)$.*
- *(Separation) $N_{\mathrm{IP}}/N_{\mathrm{LNU}} = \Omega(\varepsilon D/(k\ell\log(D/(k\ell))))$, unbounded in D.*

Interpretation. Adding irrelevant features costs LNU logarithmically (through $\log D$) but costs inner-product models linearly (through D). The gap grows without bound: at fixed k, ℓ, doubling D adds one sample for LNU but doubles the requirement for MLP.

Proof. Full proof in Appendix A.12. Upper bound: Sauer–Shelah (Sauer, 1972) on sparsity patterns + VC PAC bound + Lemma A.1 (finite-β approximation). Lower bound: standard VC lower bound (Ehrenfeucht et al., 1989).

Scope. For linear threshold models the $\Omega(D)$ lower bound is exact (VC-dimension D). We do not prove it for multi-layer ReLU networks: their large VC-dimension ($O(pL\log p)$ (Bartlett et al., 2019)) is consistent with, but does not by itself establish, an $\Omega(D)$ sample requirement on this specific sparse-DNF family. Empirically the same near-linear sample growth appears for MLPs, Transformers, and gated FFNs (§5.1). Proposition 3's gradient sparsity further gives an *implicit* bias toward relevant coordinates (non-argmin weights get gradient $\le e^{-\beta g}$), without explicit ℓ_0 regularization (Natarajan, 1995).

Statistical, not representational. The three claims are distinct. **Representationally**, Aug(1L)+FC expresses any k-term DNF (Observation 1). **Statistically**, Theorem 4 is a sample-complexity gap, not a claim that linear models cannot express the target: its lower bound is witnessed by a single conjunction $\bigwedge_{i\in S} x_i$, linearly separable as $w_S^\top x \ge |S| - \frac{1}{2}$ (Appendix A.12), so the $\Omega(D)$ cost is identifying which $k\ell$ of D features are relevant, not an inability to express. **Optimizationally**, a formal convergence guarantee is open, but SGD empirically recovers the sparse support wherever the model is accurate (Appendix J.1; correlation 0.91 over 120 runs).

Additional observations (perturbation bounds, gradient sparsity, formula extraction, norm–sparsity tradeoff) are in Appendix B.

5 Experiments

The theory (§4) predicts LNU is D-invariant while inner-product models degrade. We test this and three questions: whether the D-invariance holds and transfers to real data (§5.1); whether LNU is distractor/noise-robust and interpretable (§5.2); and whether it integrates into a Mixture-of-Experts LM (§5.4). Scope boundaries and extended benchmarks are in the appendix.

Setup. `MLP(`H`)`: one-hidden-layer ReLU MLP ($D\to H\to 1$). `Gated(`H`)`: SiLU-gated FFN (Shazeer, 2020). `Transformer(`d`,`L`)`: L-layer Transformer encoder (Vaswani et al., 2017) with model dimension d. `LNU(`H`)`: Aug(1L)+FC with H soft-AND and H soft-OR units, β=10, complement augmentation $[x, 1-x]$. XGBoost (Chen & Guestrin, 2016) comparison in Appendix J.14. **Training**: AdamW (Loshchilov & Hutter, 2019), 300 epochs, learning rate swept per method (best reported). **Evaluation**: best validation accuracy on 500 held-out samples (same criterion for all methods; last-10-epoch averages within 1pp). All benchmarks use 10 seeds unless noted (mean±std); full hyperparameters in Appendix N.

5.1 D-invariance: from theory to practice

Theorem 4 predicts that LNU needs $O(k\ell\log D)$ samples while inner-product models need $\Omega(D)$. We test this on five canonical Boolean formulas at $D \in \{20, 50, 100, 200\}$, N=100, comparing all four neural primitives (Table 1; full seven-formula results in Appendix J.7). **LNU retains 93–100% on all non-parity formulas at D=200 while all three alternatives degrade to 59–78%.** The gap is from softmin, not parameters: MLP(32) at matched param count (≈6.4k) matches MLP(16). Tree ensembles (XGBoost) also fit these clean targets; LNU's edge over them is on real, distractor-

Table 1: Five methods on canonical Boolean formulas (D=200, N=100) and real-world datasets (Dua & Graff, 2017) (N=100), 10 seeds, mean±std (%). LNU leads all four real-world (marginally vs XGBoost on Mushroom/Phishing); XGBoost wins the clean synthetic tasks.

	Synthetic Boolean (D=200)					*Real-world*			
Method	AND	OR	DNF	MAJ	XOR†	Splice	Chess	Mush.	Phish.
MLP	74.72±1.04	76.14±1.58	60.74±3.19	59.22±2.08	53.38±0.78	87.4±1.8	89.2±1.9	96.4±1.4	89.6±1.2
Gated	75.84±1.13	77.18±1.44	64.22±3.20	66.22±2.25	53.78±1.22	83.2±2.0	88.5±2.7	97.0±1.2	86.8±1.7
Transf.	76.56±1.13	78.44±1.90	64.90±3.57	65.20±3.38	53.14±0.87	37.3±2.3	91.6±1.7	94.4±1.5	89.5±2.2
XGB	100.0±0.00	100.0±0.00	99.56±0.67	100.0±0.00	62.94±17.3	91.2±1.5	92.4±1.8	96.9±1.3	90.5±1.1
LNU	**100.0**±0.00	**100.0**±0.00	92.82±7.18	98.02±1.86	56.62±4.65	**95.4**±1.5	**93.9**±1.5	**97.5**±1.3	**91.3**±0.9

Gated = SiLU-gated FFN (Shazeer, 2020). Transf. = 2-layer encoder (Vaswani et al., 2017). XGB = XGBoost (Chen & Guestrin, 2016). †1L LNU fails XOR; 2L+LayerNorm: **100%** at $D \leq 50$ (§5.5). XGB wins the clean synthetic tasks; LNU leads all 4 real-world (marginally on Mushroom/Phishing).

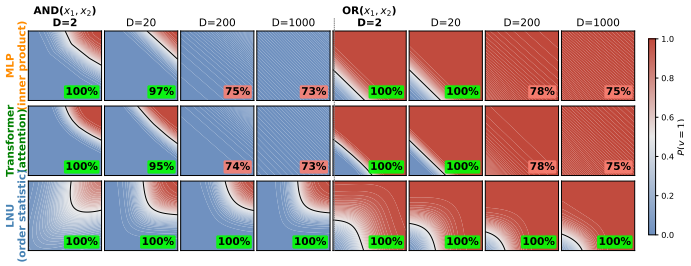


Figure 2: Decision boundaries for AND (left) and OR (right) with $D-2$ irrelevant distractors. Rows: MLP, Transformer, LNU. Badges show *validation* accuracy. MLP/Transformer degrade to 73–78% at $D \geq 200$; LNU retains 100% at all D (Proposition 3). See Figure 4 in the appendix for a geometric visualization of how distractors extend the boundary as a wall along irrelevant dimensions.

heavy, and continuous data (§5.5) and in being a differentiable, interpretable drop-in. Convergence, D-scaling, sample efficiency, noise, and feature-selection dynamics are in Appendix J.24.

Scaling law (Figure 3): $N_{\text{LNU}} \approx 50$ (constant across D) vs $N_{\text{MLP}} \approx 1.2 \cdot D^{0.97}$ (Mann–Whitney $p > 0.36$ for LNU D-invariance; paired Wilcoxon $p = 0.001$ for LNU > MLP). The near-linear growth $N_{\text{MLP}} \propto D^{0.97}$ confirms Theorem 4's lower bound $N_{\text{IP}} = \Omega(D)$, while LNU's constant N_{crit} is consistent with $O(k\ell \log D)$. The same D-degradation holds for the Gated-MLP and Transformer baselines (Appendix J.24, Figure 10), consistent with all weighted-sum primitives diluting at $O(1/\sqrt{D})$. At D=2000, N=200: LNU reaches 100% on AND/MAJ while MLP plateaus at 56–75% (Appendix J.23).

Statistical validation on random formulas. Beyond canonical formulas, LNU's advantage holds on 30 random DNF formulas (2–4 terms, 2–4 literals, D=100, N=100, 10 seeds): LNU 81.2%, MLP 72.6%, Transformer 70.1%. Wilcoxon: LNU vs MLP $p = 6.9\times10^{-5}$; LNU vs Transformer $p = 2.3\times10^{-6}$. With Holm correction, **19/30 formulas significant** at α=0.05 (Appendix J.8).

Prior aggregators. At D=100: NLN and LNN reach only 75.1% on AND (near chance) vs LNU 100.0%, confirming Propositions 1, 2. A gradient-trained lookup-table network (DWN (Bacellar et al., 2024)), under the same protocol, trails LNU on every canonical target and collapses under distractors (Appendix J.3, Table 12), placing the trainable-LUT family on the learning axis too (§2).

5.2 Real-world datasets, robustness, and interpretability

The synthetic results raise a natural question: does LNU's advantage transfer to real data? We test on four UCI datasets with known logical target structure (Table 1, right columns).

LNU wins on all four (Splice +8.0pp, Chess +4.7pp, Mushroom +1.1pp, Phishing +1.7pp over MLP). The advantage narrows at larger N (e.g. +1.6pp Chess, +0.4pp Mushroom at N=200; Appendix J.15).

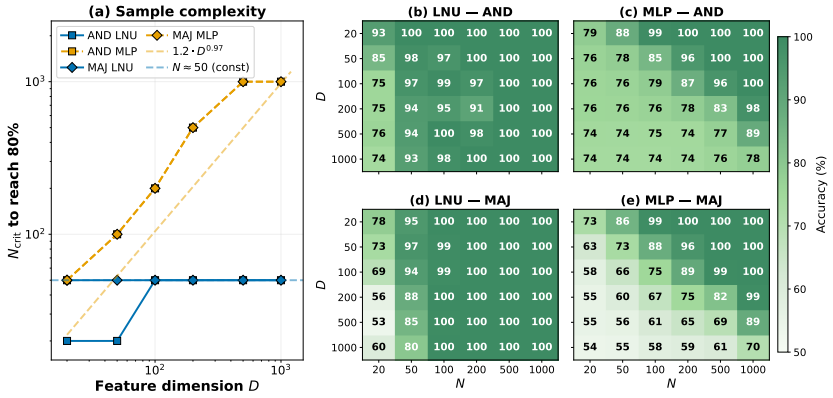


Figure 3: **(a)** N_{crit} (samples to reach 80%) vs D: LNU constant at $\approx$50; MLP grows as $1.2 \cdot D^{0.97}$. **(b–e)** Accuracy heatmaps ($D \times N$) for AND and MAJ: LNU reaches 100% at N=100 regardless of D; MLP plateaus at 74–78% (AND) or degrades to 54–60% (MAJ). 10 seeds.

Table 2: Distractor robustness: accuracy (%) as irrelevant features are padded. LNU is near-invariant; Transformer collapses worst. Splice N=100; 20News N=200; 10 seeds.

D	MLP	Transf.	**LNU**
Splice (DNA), N=100:			
287	87.4±2.1	37.3±2.3	**95.4**±1.5
5k	65.6±0.8	10.9±1.1	**94.8**±1.6
20 Newsgroups (BoW), N=200:			
5k	90.1±1.4	**97.4**±1.2	81.7±4.7
20k	70.4±2.6	65.6±3.1	**71.4**±3.3
50k	65.3±3.4	52.5±2.8	**71.8**±5.2

Table 3: Label-noise robustness (N=100, 10 seeds). $\Delta_{\text{LNU}-\text{MLP}}$ in pp±std. LNU leads at 0–10% noise on all four datasets. At 20%, MLP overtakes on Mushroom and Phishing, consistent with inner-product averaging acting as implicit regularization against label corruption; LNU still leads on Splice and Chess.

	$\Delta_{\text{LNU}-\text{MLP}}$ (pp)		
Dataset (D)	0%	10%	20%
Splice (287)	+8.0±2.3	+6.2±2.5	+3.1±3.0
Chess (73)	+4.7±2.4	+4.0±2.6	+2.1±2.8
Mushroom (117)	+1.1±1.9	+0.8±2.1	−2.3±2.4
Phishing (30)	+1.7±1.5	+1.2±1.7	−4.8±2.0

Distractor robustness. Theorem 4 predicts that LNU should be insensitive to irrelevant features. On Splice (Noordewier et al., 1991) padded from D=287 to 5,000: **LNU** −0.6**pp, MLP** −21.8**pp, Transformer** −26.4**pp** (Table 2, Figure 2). On 20 Newsgroups (Lang, 1995) BoW, LNU overtakes MLP only under heavy padding (+6.5pp at D=50,000, but trailing at moderate D; Table 2), 10 seeds; Appendix J.16. LNU also leads MLP at 0–10% label noise (Table 3), with MLP overtaking on two datasets at 20%.

Interpretability. A key advantage of softmin: LNU's weights α_i directly reveal which features each unit selects. On AND(x_1, x_2) at D=200: LNU reaches 95% of total gradient importance with k=56 features; MLP requires k=150 (3× sparser). Feature knockout confirms causality: +26pp gap (Figure 6). The pattern holds across all four formulas (Figure 7 in the appendix).

Further results and controls. The appendix adds noise robustness (Appendix J.10), an ℓ_1-regularized MLP control (Appendix J.12), continuous-vs-binary LNU (Appendix J.17), FOLIO NLI (Appendix J.19), the distractor-crossover hybrid (Appendix J.20), extended benchmarks (Appendix J.22), OLMoE details (Appendix J.18), and the RuleTaker limitation (Appendix H).

5.3 Continuous Inputs

LNU is not restricted to binary inputs: continuous features pass through a sigmoid step (§3) into the softmin/softmax range, the mechanism otherwise unchanged. On the seven canonical formulas with *Gaussian* inputs (each coordinate thresholded to define a literal, D=200, N=200, 10 seeds), LNU wins six of seven by +7 to +22pp over the strongest baseline (full table in Appendix J.5, Table 14);

the exception is XOR. This is the property the LUT/logic-gate family cannot share (§2): they require discretized inputs, whereas LNU consumes continuous inputs directly.

5.4 LNU AS A COMPOSABLE EXPERT IN A MIXTURE-OF-EXPERTS MODEL

As a feasibility check that the Boolean bias survives at scale, we use LNU as an expert in a Mixture-of-Experts (MoE) model. Dropped into a pretrained OLMoE-1B-7B (Muennighoff et al., 2024) (2 of 64 experts), LNU matches the FFN expert at 31% fewer per-expert parameters; a from-scratch hybrid (4FFN+4LNU) outperforms all-FFN at 14% fewer parameters; and on FOLIO (Han et al., 2022) LNU shows a few-shot advantage ($+4.2$pp at $N{=}100$). These results are exploratory (3 seeds, high variance), so we position LNU as a *composable logic-specialist expert* alongside standard FFN experts, not a wholesale replacement. Full tables and the from-scratch study are in Appendix J.6.

5.5 CONTROLS, SCOPE, AND POSITIONING

Two distinct advantages, not conflated. Theorem 4's separation is purely *statistical*: its witness (a single conjunction) is linearly separable, so the $\Omega(D)$ cost is identifying the relevant features, not expressing the target (§4). A *separate*, representational advantage appears for targets that are *not* linearly separable: on a two-term DNF (a union of conjunctions), sparse and linear baselines (ℓ_1-logistic regression, ℓ_0 best-subset, ElasticNet, univariate screening) cannot exceed 91% *training* accuracy even at $N{=}2000$, whereas Aug(1L)+FC fits it exactly (100%) and no regularization closes the gap. The two are complementary: the sample-complexity separation holds on linearly-separable targets where expressivity is not at issue, while this control isolates a representational gap on non-separable DNF. We do not conflate them: Theorem 4 is not a representational claim.

Multi-layer composition and scope boundaries. Single-layer LNU computes one level of Boolean logic (DNF); deeper targets require stacking, and a two-layer LNU with **LayerNorm** connectors solves XOR at 100% ($D{\leq}50$) versus 91%/61% for STE/sigmoid connectors (Appendix K).

Scope boundaries. LNU helps only when the target has sparse threshold-Boolean structure (binary or continuous, §5.3); where that structure is absent it does not, which we report: relational reasoning (bAbI (Weston et al., 2015), Transformer wins), semantic text (native 20 Newsgroups (Lang, 1995), MLP $+8.4$pp), and dose-dependent targets (Tox21 (Wu et al., 2018), MLP ties).

Versus other logical methods. Tree ensembles match LNU on clean synthetic targets but are neither differentiable nor distractor-robust; trainable-logic/LUT methods and a RIPPER rule learner collapse under distractors and must binarize continuous inputs, where LNU leads by $+4$ to $+10$pp (Appendix J.2, Table 13).

When LNU helps. The pattern is consistent across these experiments: LNU wins most where the theory predicts, sparse Boolean targets in high D, low N, and with distractors (binary or continuous), and its advantage narrows or vanishes outside that regime.

6 CONCLUSION

Modern networks aggregate by weighted sums, but the operations at the core of logic, AND and OR, are extrema, not sums. When a prediction target is a Boolean function of a few relevant features, the sparse, logically structured regime where tree and rule learners remain competitive with deep networks on tabular and scientific data, that mismatch is costly, and extremal selection by softmin/softmax removes it: it provably needs fewer samples than a weighted sum, a PAC separation of $O(k\ell \log D)$ versus $\Omega(D)$ (Theorem 4). Empirically LNU wins where that structure is present (92–100% on non-parity Boolean formulas, near-invariance to distractors) and loses where it is not, which we report. The Logical Neural Unit is thus a drop-in, differentiable, continuous-input primitive that complements the inner product rather than replacing it: a structural inductive bias, useful exactly when a target is Boolean-structured. The main open problem is a formal optimization guarantee (SGD recovers the sparse support empirically, Appendix J.1, but without proof); next steps are learnable per-unit β (§M.1) and LNU-native pretraining at scale.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, namely writing and debugging our experiment and analysis code (a task with required disclosure). We have reviewed all AI-assisted work and take responsibility for the final content of this paper.

ETHICS STATEMENT

This work introduces a general-purpose differentiable learning primitive and evaluates it on synthetic Boolean tasks and standard public benchmarks; it involves no human subjects and no personal, sensitive, or private data, and we foresee no direct risk of harm specific to the method. If anything, LNU's polynomial-time formula extraction (§5.2) improves model transparency and auditability relative to opaque inner-product baselines, which may aid deployment in settings that require interpretable decisions.

REPRODUCIBILITY STATEMENT

The LNU primitive is fully specified by the equations in §3 (softmin/softmax with sharpness β over the complement-augmented input $[x, 1-x]$, then a linear read-out) and can be implemented from them in a few lines. All theoretical results state their assumptions explicitly, with complete proofs in Appendix A. The full experimental protocol (data generation, train/validation splits, seeds, architectures, and hyperparameters) is given in §5 and Appendix N, so every experiment is reproducible from the paper alone. All reported numbers are means over multiple seeded runs with standard deviations, using the same seeds, splits, and budget across methods; each run draws its data from a seeded generator, so subsets are exactly reconstructable (Appendix N). We will release the full code upon acceptance.

REFERENCES

Marta Andronic and George A. Constantinides. PolyLUT: Learning piecewise polynomials for ultra-low latency FPGA LUT-based inference. In *International Conference on Field Programmable Technology (ICFPT)*, 2023.

Marta Andronic and George A. Constantinides. NeuraLUT: Hiding neural network density in boolean synthesizable functions. In *International Conference on Field-Programmable Logic and Applications (FPL)*, 2024.

Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

Alan Tendler Leibel Bacellar, Zachary Susskind, Mauricio Breternitz Jr, Eugene John, Lizy Kurian John, Priscila Machado Vieira Lima, and Felipe M. G. França. Differentiable weightless neural networks. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Peter L. Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.

Adrien Benamira, Thomas Peyrin, Trevor Yap, Tristan Guérand, and Bryan Hooi. Truth table net: Scalable, compact and verifiable neural networks with a dual convolutional small boolean circuit networks form. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD*, pp. 785–794, 2016.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In *Proceedings of IJCAI*, 2020.

William W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML)*, pp. 115–123, 1995.

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. In *International Conference on Learning Representations*, 2019.

Dheeru Dua and Casey Graff. UCI machine learning repository. https://archive.ics.uci.edu/ml, 2017.

Andrzej Ehrenfeucht, David Haussler, Michael Kearns, and Leslie Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–261, 1989.

Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022.

Petr Hájek. *Metamathematics of Fuzzy Logic*, volume 4 of *Trends in Logic*. Kluwer Academic Publishers, 1998.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, et al. FOLIO: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*, 2022.

Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pp. 6–20, 1986.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

Ken Lang. NewsWeeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pp. 331–339, 1995.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

André F. T. Martins and Ramón Fernández Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pp. 1614–1623, 2016.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. OLMoE: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*, 2024.

Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

Michiel O. Noordewier, Geoffrey G. Towell, and Jude W. Shavlik. Training knowledge-based neural networks to recognize genes in DNA sequences. In *Advances in Neural Information Processing Systems*, volume 3, pp. 530–536, 1991.

Ali Payani and Faramarz Fekri. Learning algorithms via neural logic networks. *arXiv preprint arXiv:1904.01554*, 2019.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Deep differentiable logic gate networks. In *Advances in Neural Information Processing Systems*, 2022.

Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh Srivastava. Logical neural networks. *arXiv preprint arXiv:2006.13155*, 2020.

Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1): 145–147, 1972.

Luciano Serafini and Artur S. d'Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. In *NeSy Workshop*, 2016.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of ACL*, 2021.

Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. FINN: A framework for fast, scalable binarized neural network inference. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2017.

Yaman Umuroglu, Yash Akhauri, Nicholas J. Fraser, and Michaela Blott. LogicNets: Co-designed neural networks and circuits for extreme-throughput applications. In *International Conference on Field-Programmable Logic and Applications (FPL)*, 2020.

Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Erwei Wang, James J. Davis, Peter Y. K. Cheung, and George A. Constantinides. LUTNet: Rethinking inference in FPGA soft logic. In *IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2019.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-Complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, 9:513–530, 2018.

Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision-making. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):183–190, 1988.

Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

Appendix Overview

The appendix is organized as follows.

Theory. Appendix A gives the full proofs (the finite-β softmin tail bound, Propositions 1–3, and Theorem 4, the PAC separation); Appendix B collects the supporting observations (perturbation bound, gradient sparsity, formula extraction, norm–sparsity tradeoff).

Mechanism and ablations (what makes the primitive work). Appendix C (mechanism and distractor geometry), then the architectural ablations: Appendix D (FC vs. LNU-head readout), Appendix E (AND-only vs. OR-only vs. both units), Appendix F (complement augmentation), and Appendix G (sigmoid-preprocessing MLP control).

Scope and positioning. Appendix H documents a bag-of-words feature limitation, and Appendix I gives extended related work and the five-desiderata primitive comparison.

Extended experiments. Appendix J collects the additional results referenced from §5: the MNIST and trainable-logic/LUT comparison, the on-axis DWN/RIPPER study, continuous inputs, the OL-MoE expert swap, real-world tabular learning curves, and D-scaling. Appendix K reports the multi-layer, compositional study (XOR).

Open problems, discussion, and reproducibility. Appendix L lists open problems; Appendix M discusses implications; and Appendix N gives the full experimental details for reproducibility.

A Formal Proofs and Architectural Ablations

This appendix contains: **A.1–A.12**: formal proofs of Lemma A.1 and the observations, propositions, and theorem stated in the main text. **A.13**: supporting observation statements. **B–G**: architectural ablations (readout head, AND/OR, sigmoid control, augmentation). **H**: extended related work.

We collect formal proofs of the observations stated in Section 4. Throughout, $\mathrm{Aug}(x) = [x, 1-x] \in [0,1]^{2D}$, and the j-th soft-AND unit computes $\mathrm{AND}_j(x) = \sum_{i=1}^{2D} \alpha_{ji}(x) z_{ji}(x)$ with $z_{ji}(x) = \mathrm{Aug}(x)_i \cdot w_{ji}^{\mathrm{AND}}$ and $\alpha_{ji}(x) = \mathrm{softmax}(-\beta z_j)_i$.

A.1 Lemma A.1 (Softmin tail bound on order statistics)

Statement. Let $z = (z_1, \ldots, z_n) \in \mathbb{R}^n$ with $|z_i| \le M$, and let $z_{(1)} \le z_{(2)} \le \ldots \le z_{(n)}$ be the order statistics. Define the distinct minimum gap $g := z_{(r+1)} - z_{(1)}$, where r is the multiplicity of the minimum. Then for $\beta > 0$,

$$0 \le \mathrm{softmin}_\beta(z) - z_{(1)} \le \frac{2M(n-r)}{r}\, e^{-\beta g}.$$

Proof. Let $S^\star = \{i : z_i = z_{(1)}\}$ with $|S^\star| = r$, and $S^\circ = [n] \setminus S^\star$. For $i \in S^\star$, $\mathrm{softmax}(-\beta z)_i = 1/(\sum_k e^{-\beta(z_k - z_{(1)})})$. For $k \in S^\circ$, $z_k - z_{(1)} \ge g$, so $e^{-\beta(z_k - z_{(1)})} \le e^{-\beta g}$. The denominator is in $[r, r + (n-r)e^{-\beta g}]$, giving $\mathrm{softmax}(-\beta z)_i \in [1/(r + (n-r)e^{-\beta g}), 1/r]$ for $i \in S^\star$.

Rewrite the softmin output as $\mathrm{softmin}_\beta(z) - z_{(1)} = \sum_{k \in S^\circ} \mathrm{sm}_k (z_k - z_{(1)})$. For $k \in S^\circ$, $\mathrm{sm}_k = e^{-\beta z_k}/Z$ where $Z \ge r \cdot e^{-\beta z_{(1)}}$, so $\mathrm{sm}_k \le e^{-\beta(z_k - z_{(1)})}/r \le e^{-\beta g}/r$. Combined with $z_k - z_{(1)} \le 2M$, each summand is $\le 2Me^{-\beta g}/r$ and there are $(n-r)$ of them. Hence the bound. If $r = n$ (all values tied) then $\mathrm{softmin}_\beta(z) = z_{(1)}$ exactly. □

A.2 Proof of Observation 1 (Functional completeness)

Architectural caveat. Section 3's LNU primitive computes $z_{ji} = \mathrm{Aug}(x)_i \cdot w_{ji}^{\mathrm{AND}}$ with no bias inside the softmin. As a consequence, every $x \in \{0,1\}^D$ has D entries of $\mathrm{Aug}(x)$ equal to zero, forcing $z_{ji} = 0$ on those entries. We construct f as a linear combination of single-literal soft-AND features through the FC layer.

Construction (literal-extracting AND units + FC).

Step 1: literal extractor. For each literal in the DNF, allocate one AND unit with a single nonzero weight at the appropriate Aug position (set to -1).

Step 2: literal feature value. By Lemma A.1, the soft-AND output equals -1 when the literal is satisfied (gap g=1; error $\le 2(2D-1)e^{-\beta}$) and equals 0 when not satisfied.

Step 3: FC composition (two-layer ReLU head). For each minterm t with s_t literals, let $S_t(x) = \sum_{j=1}^{s_t} u_{t,j}(x)$ be the sum of the s_t literal outputs (each ≈ -1 when satisfied, 0 otherwise). Define $h_t(x) = \mathrm{ReLU}(-S_t(x) - (s_t - 1))$ (equals $\approx$1 iff all s_t literals are satisfied) and $y(x) = 1 - \mathrm{ReLU}(1 - \sum_t h_t(x))$ (equals 1 iff any minterm fires).

Step 4: error control. Each $u_{t,j}$ has approximation error $\varepsilon_\beta = 2(2D-1)e^{-\beta}$. Total network error $\le k s_{\max} \varepsilon_\beta$. Choose $\beta \ge \log(4Dks_{\max}/\eta)$. □

Two honest concessions. (i) The construction uses $H = \sum_t s_t \le k\ell$ AND units to extract individual literals. (ii) The construction relies on a multi-layer FC head (two-layer ReLU); a single linear FC layer suffices only for restricted classes.

A.3 Proof of Observation 2 (Perturbation bound)

Proof. By the chain rule,

$$\nabla_x \mathrm{AND}_j(x) = \sum_i \alpha_i \nabla_x z_{ji} + \sum_i z_{ji} \nabla_x \alpha_i.$$

For the softmin selection weights, $\nabla_x \alpha_i = -\beta \alpha_i (\nabla_x z_{ji} - \sum_k \alpha_k \nabla_x z_{jk})$. Substituting,

$$\nabla_x \mathrm{AND}_j = \sum_i \alpha_i \nabla_x z_{ji} - \beta \sum_i \alpha_i (z_{ji} - \bar{z}_j) \nabla_x z_{ji}.$$

Taking ℓ_∞ norm and using $|\partial z_{ji}/\partial x_k| \le |w_{ji}|$ and $|z_{ji} - \bar{z}_j| \le 2M$,

$$\left| \frac{\partial \mathrm{AND}_j}{\partial x_k} \right| \le (1 + 2\beta M)(|w_{jk}| + |w_{j,D+k}|).$$

Summing over $k \in [D]$ gives $\|\nabla_x \mathrm{AND}_j\|_1 \le (1 + 2\beta M)\|w_j\|_1$. By the mean-value theorem with Hölder's inequality, $|\mathrm{AND}_j(x+\delta) - \mathrm{AND}_j(x)| \le \varepsilon(1 + 2\beta M)\|w_j\|_1$. □

A.4 Proof of Observation 3 (Gradient sparsity)

Proof. $\mathrm{AND}_j = \sum_k \alpha_k z_{jk}$ with $\alpha_k = e^{-\beta z_{jk}}/Z$. Differentiating,

$$\frac{\partial \mathrm{AND}_j}{\partial z_{ji}} = \alpha_i + \sum_k z_{jk} \frac{\partial \alpha_k}{\partial z_{ji}} = \alpha_i (1 - \beta(z_{ji} - \bar{z}_j)).$$

For $z_{ji} > z_{(1)} + \eta$, $\alpha_i \le e^{-\beta(z_{ji} - z_{(1)})} \le e^{-\beta\eta}$, and $|1 - \beta(z_{ji} - \bar{z}_j)| \le 1 + 2\beta M$. Hence $|\partial \mathrm{AND}_j / \partial z_{ji}| \le (1 + 2\beta M) e^{-\beta\eta}$. □

A.5 Proof of Observation 4 (Formula extraction)

Dominance condition. The LNU layer satisfies the (τ, η)-dominance condition over input set $\mathcal{X}$ if there exist constants $\tau > 0, \eta > 0$ and a function $i_j^* : \mathcal{X} \to [2D]$ such that for every $x \in \mathcal{X}$ and every unit j: (D1) $|W^{\mathrm{AND}}_{j,i_j^*(x)}| \ge \tau$; (D2) for every other coordinate $i \ne i_j^*(x)$ with $|W^{\mathrm{AND}}_{j,i}| > \tau$, $z_{j,i}(x) \ge z_{j,i_j^*(x)}(x) + \eta$.

Claim. If the LNU layer satisfies (τ, η)-dominance on $\mathcal{X}$ and the network's output at $\beta{=}\infty$ is correct on $\mathcal{X}$, then the extracted formula $\tilde{f}(x) := \bigvee_j v_j \tilde{C}_j(x)$ agrees with the network output on $\mathcal{X}$ provided $\beta \ge \beta^* := \log(4D/\tau)/\eta$.

Proof sketch. By Lemma A.1 and the dominance hypothesis, $\alpha_{j,i_j^*(x)}(x) \ge 1 - 2De^{-\beta\eta}$, so the soft-AND output is within $O(\varepsilon)$ of the dominant literal's weighted value. □

A.6 Proof of Observation 5 (Nonzero-support bound)

Proof. The literal-extractor construction in Appendix A.2 allocates one AND unit per literal in the DNF, with exactly one nonzero weight per unit. Total AND-layer nonzero weights: $\sum_t s_t \le k\ell$, independent of D. The FC composition adds $O(ks_{\max})$ weights. □

A.7 Proof of Proposition 1 (NLN product underflow)

Proof. Each of ρD large-set factors is $\leq 1-\tau$; the remaining factors are ≤ 1. Since every factor is in $[0,1]$, the product is monotone: $\prod \leq (1-\tau)^{\rho D}$. For the gradient, $\partial \mathrm{AND}_{\mathrm{NLN}}/\partial x_j = w_j \prod_{i \neq j}(1 - w_i(1-x_i))$, bounded by $(1-\tau)^{\rho D - 1}$. □

Generic justification. Under random initialization with $w_i, x_i \sim U[0,1]$ independent, $\Pr[w_i(1 - x_i) \geq 1/4] \approx 0.40$. By Hoeffding's inequality (Hoeffding, 1963), $\Pr[\#\{i : w_i(1-x_i) \geq 1/4\} \geq \rho D] \geq 1 - e^{-2(p-\rho)^2 D}$. Taking $\rho = 0.3$ gives the hypothesis with probability $\geq 1 - e^{-0.02D}$.

A.8 Proof of Proposition 2 (LNN threshold concentration)

Proof. $S = \sum w_i x_i$ is a sum of D independent Bernoulli$(1/2)$-scaled variables. Linearity gives $\mathbb{E}[S] = \sum w_i/2$ and $\mathrm{Var}[S] = \sum w_i^2/4 \in [c^2 D/4, C^2 D/4]$. Single-coordinate sensitivity $|\partial S/\partial x_j| = |w_j| \leq C$. Dividing by $\sqrt{\mathrm{Var}[S]} \geq c\sqrt{D}/2$ gives $|\partial S/\partial x_j|/\sqrt{\mathrm{Var}[S]} \leq 2C/(c\sqrt{D}) = O(1/\sqrt{D})$. □

A.9 Proof of Proposition 3 (LNU gap-dependent approximation)

Proof.

Step 1 (Approximation bound). Apply Lemma A.1 with the n entries of z ($n = 2D$ for Aug(1L)+FC operating on $\mathrm{Aug}(x) \in [0,1]^{2D}$). Lemma A.1 gives $0 \leq \mathrm{AND}_{\mathrm{LNU}}(x) - z_{(1)} \leq \frac{2M(n-r(x))}{r(x)} e^{-\beta g(x)}$.

Step 2 (D-independent single-coordinate sensitivity). The softmin selection weight on the argmin coordinate i^* satisfies $\alpha_{i^*} \to 1/r$ as $\beta \to \infty$. The derivative $\partial \mathrm{AND}_{\mathrm{LNU}}/\partial z_{i^*} = \alpha_{i^*}(1 - \beta(z_{i^*} - \bar{z}))$ where $\bar{z} = \mathrm{AND}_{\mathrm{LNU}}$. At large β with $i^* = \mathrm{argmin}$, $z_{i^*} \to \bar{z}$, so the parenthetical $\to 1$. Hence $\partial \mathrm{AND}_{\mathrm{LNU}}/\partial x_{i^*} = w_{i^*}(1/r + O(ne^{-\beta g_0})) = \Theta(|w_{i^*}|)$ where $g_0 := \inf_x g(x)$ is the minimum gap, with no polynomial dependence on D. □

Contrast. Proposition 1 (NLN) decays in D at rate $(1-\tau)^{\rho D}$. Proposition 2 (LNN) loses sensitivity at rate $O(1/\sqrt{D})$. Proposition 3 (LNU) retains $\Theta(|w_j|)$ sensitivity with error exponentially small in βg, not in D. This is the separation the paper proves.

A.10 Proof of Observation 6 (Norm–sparsity tradeoff)

Proof (set-separation argument). Let $L(x) := w^\top x + b$. Decompose $L = L_S + L_{\bar{S}}$ with L_S deterministic given x_S and $L_{\bar{S}}$ random.

Step 1: Hoeffding on $L_{\bar{S}}$. For $x_{\bar{S}} \sim \mathrm{Bernoulli}(1/2)^{D-|S|}$, $Z := \sum_{i \in \bar{S}} w_i(x_i - 1/2)$. Hoeffding: $\Pr[|Z| > t] \leq 2\exp(-2t^2/\|w_{\bar{S}}\|_2^2)$. Setting RHS to δ gives $t(\delta) = \|w_{\bar{S}}\|_2 \sqrt{\log(2/\delta)/2}$.

Step 2: Set separation. Since f is non-constant on $\{0,1\}^{|S|}$, there exist x_S^+, x_S^- with $f(x_S^+) = 1$, $f(x_S^-) = 0$. Correct classification with prob $\geq 1 - \delta$ requires $L_S(x_S^+) - L_S(x_S^-) > 2t(\delta)$.

Step 3: Upper bound. $L_S(x_S^+) - L_S(x_S^-) = w_S^\top(x_S^+ - x_S^-)$. By Hölder, $|w_S^\top(x_S^+ - x_S^-)| \leq \|w_S\|_1$.

Step 4: Combine. $\|w_S\|_1 > 2t(\delta) = \|w_{\bar{S}}\|_2 \sqrt{2\log(2/\delta)}$. Under uniform-floor, $\|w_{\bar{S}}\|_2 \geq W\sqrt{\|w\|_0^{\mathrm{irr}}}$, so $\|w_S\|_1 > W\sqrt{2\|w\|_0^{\mathrm{irr}} \log(2/\delta)}$. □

A.11 Why Aug(1L)+FC does not face the bound of Observation 6

Aug(1L)+FC is not a depth-one linear classifier: its first layer is a softmin/softmax nonlinearity that performs per-example softmin-weighted feature selection (Observation 3). The soft-AND output is a data-dependent convex combination rather than a fixed-weight linear map. In the signed-weight construction of Appendix A.2, the irrelevant-feature weights are literally 0, so the bound's premise $\|w\|_0^{\mathrm{irr}} = 0$ is satisfied (case (ii) of Observation 6 applies).

A.12 THEOREM 4: PAC SAMPLE-COMPLEXITY SEPARATION

We promote the norm–sparsity tradeoff to a sample-complexity separation. The upper bound combines Lemma A.1 with the Aug(1L)+FC sparsity-support claim and a Sauer–Shelah (Sauer, 1972) union bound.

Setup. Fix $k, \ell \geq 1$ and let $\mathcal{F}_{k,\ell,D}$ be Boolean functions over $\{0,1\}^D$ expressible as DNF with at most k terms of at most ℓ literals each. Let $\mathcal{D}$ be the uniform distribution over $\{0,1\}^D$. We say $\mathcal{H}$ **PAC-learns** $\mathcal{F}_{k,\ell,D}$ with sample complexity $N(\varepsilon, \delta)$ if there exists an algorithm that, given $N(\varepsilon, \delta)$ i.i.d. samples labeled by any $f \in \mathcal{F}_{k,\ell,D}$, returns $\hat{h} \in \mathcal{H}$ satisfying $\Pr[L(\hat{h}) \leq \varepsilon] \geq 1 - \delta$.

Theorem 4. As stated in Section 4.2.

Proof of (a): upper bound. *Step A.1 (Realizability).* By Observation 5 (Appendix A.6), every $f \in \mathcal{F}_{k,\ell,D}$ is exactly representable by an Aug(1L)+FC network in $\mathcal{H}^s_{\mathrm{LNU}}$ at $\beta = \infty$ with $H = k\ell$ AND units and $\|W^{\mathrm{AND}}\|_0 = k\ell$.

Step A.2 (Finite-β approximation). By Lemma A.1, each soft-AND output differs from its hard-min limit by at most $2(2D-1)e^{-\beta}$. Compositional error through $H = k\ell$ AND units and the head is at most $4Dk\ell\, e^{-\beta}$. Choosing $\beta^* \geq \log(8Dk\ell/\varepsilon)$ keeps this below $\varepsilon/2$.

Step A.3 (Hypothesis-class size). Each AND unit selects its active input coordinates from the $2D$ augmented dimensions. The sparsity constraint $\|W^{\mathrm{AND}}\|_0 \leq s$ means the network uses at most s distinct input coordinates across all units. The number of such support sets is $\binom{2D}{s} \leq (2De/s)^s$. Within each fixed support, the network has $O(s)$ real parameters (s AND-layer weights + $O(s)$ FC weights). The ReLU FC head is piecewise-linear; the smooth softmin layer has VC-dimension at most that of a piecewise-linear network with the same parameter count (smooth activations have fewer decision boundaries). Applying standard VC bounds for piecewise-linear networks (Bartlett et al., 2019) gives $\mathrm{VC} = O(p \log p) = O(s \log s) \leq O(s^2)$. Taking the union over support sets,

$$\mathrm{VC}(\mathcal{H}^s_{\mathrm{LNU}}) \leq s \log(2De/s) + O(s^2) = O(k\ell \log(D/(k\ell)) + (k\ell)^2).$$

For sparse Boolean targets with $k\ell = O(\log D)$, the $s \log(D/s)$ term dominates.

Step A.4 (PAC bound). By the agnostic VC PAC bound, if $N \geq (c/\varepsilon^2)(\mathrm{VC}(\mathcal{H}^s_{\mathrm{LNU}}) + \log(1/\delta))$, then ERM returns $\hat{h}$ with $L(\hat{h}) \leq L^* + \varepsilon$ with probability $\geq 1 - \delta$. Combined with $L^* \leq \varepsilon/2$: $N = O((k\ell \log(D/(k\ell)) + (k\ell)^2 + \log(1/\delta))/\varepsilon^2)$. For $k\ell = O(\log D)$, the $(k\ell)^2$ term is absorbed. □

Proof of (b): lower bound. *Step B.1.* The class of D-variable linear threshold functions has VC-dimension $D + 1$ (Vapnik & Chervonenkis, 1971; Anthony & Bartlett, 1999).

Step B.2. The class $\mathcal{F}_{k,\ell,D}$ contains for each $S \subseteq [D]$ with $|S| \leq k\ell$ the single-conjunction function $f_S(x) = \bigwedge_{i \in S} x_i$, which is realizable in $\mathcal{H}_{\mathrm{IP}}$ as $w_S^\top x \geq |S| - 1/2$.

Step B.3. Any algorithm that PAC-learns $\mathcal{F}_{k,\ell,D}$ using hypotheses from $\mathcal{H}_{\mathrm{IP}}$ requires $N \geq \Omega(\mathrm{VC}(\mathcal{H}_{\mathrm{IP}})/\varepsilon) = \Omega((D + \log(1/\delta))/\varepsilon)$ by the realizable lower bound (Ehrenfeucht et al., 1989). (The bound holds for proper learners; improper learners can potentially circumvent it, but standard MLP training is proper w.r.t. the linear class.) □

Proof of (c): separation. $\frac{N_{\mathrm{IP}}}{N_{\mathrm{LNU}}} = \frac{\Omega(D/\varepsilon)}{O(k\ell \log(D/(k\ell))/\varepsilon^2)} = \Omega(\varepsilon D/(k\ell \log(D/(k\ell))))$, unbounded in D for fixed k, ℓ. □

Discussion.

- The separation is between hypothesis classes *with vs. without a sparsity prior*, not a fundamental impossibility for depth-one linear learning. A learner using $\mathcal{H}^s_{\mathrm{IP}} = \{w : \|w\|_0 \leq s\}$ matches the LNU upper bound. The Aug(1L)+FC architecture **builds the sparsity prior into the architecture and optimization dynamics**; explicit ℓ_0 regularization is NP-hard (Natarajan, 1995).

- The upper bound assumes computational tractability of ERM at the sparse class. ERM over $\mathcal{H}^s_{\mathrm{LNU}}$ with sparsity is in general NP-hard; in practice, gradient descent finds approximate solutions through softmin's gradient-routing property (Observation 3).

- The separation does not extend to multi-layer MLPs. The VC-dimension of an L-layer MLP with parameter count p is $O(pL \log p)$ (Bartlett et al., 2019).

B Supporting Observations

Observation 1 (Functional completeness)**.** *Any k-term DNF with at most ℓ literals per term is exactly representable by an Aug(1L)+FC network at $\beta \to \infty$, using one soft-AND unit per literal and a two-layer ReLU head for minterm composition. Since every Boolean function $f : \{0,1\}^D \to \{0,1\}$ has a DNF, this covers all such f. (Proof: Appendix A.2.)*

Observation 2 (Perturbation bound)**.** *For $\beta > 0$ and $\|\delta\|_\infty \le \varepsilon$ on inputs in $[0,1]^D$, the output of an LNU AND unit satisfies $|\mathrm{AND}_j(x+\delta) - \mathrm{AND}_j(x)| \le \varepsilon(1+2\beta M)\|w_j\|_1$. (Proof: Appendix A.3.)*

Observation 3 (Gradient sparsity)**.** *The softmin/softmax selection weights α_{ji} in an LNU layer concentrate on the extreme inputs as β increases; the gradient $\partial \mathrm{AND}_j / \partial x_i$ is proportional to α_{ji} and decays exponentially in the gap. (Proof: Appendix A.4.)*

Observation 4 (Formula extraction)**.** *At convergence, each LNU hidden unit reads off as a Boolean formula: for AND unit j, active literals are those i with $|(w_j^{\mathrm{AND}})_i|$ above threshold, with polarity determined by whether the weight references x_i or $(1 - x_i)$.*

Observation 5 (Nonzero-support bound)**.** *The literal-extracting construction of Observation 1 uses at most $\sum_t s_t \le k\ell$ nonzero AND-layer weights (with $s_t \le \ell$ the number of literals in term t), independent of D: the required capacity scales with the DNF complexity of the target, not with D. (Proof: Appendix A.6.)*

Observation 6 (Norm–sparsity tradeoff)**.** *Let $h(x) = \sigma(w^\top x + b)$ with σ strictly monotone, and assume the uniform-floor hypothesis: for every $i \notin S$ with $w_i \neq 0$, $|w_i| \ge W > 0$. To classify $f \in \mathcal{F}_{k,\ell,D}$ correctly with probability $\ge 1 - \delta$, either $\|w_S\|_1 \ge W\sqrt{2\|w\|_0^{\mathrm{irr}} \log(2/\delta)}$ or $\|w\|_0^{\mathrm{irr}} = 0$.*

Empirical signature. At D=1000, N=100, MLPs plateau at 59–76% while LNU remains at or near 100% on AND, OR, MAJ (§5.1). This is *consistent with* the representational obstruction described by Observation 6 and the sample-complexity separation of Theorem 4.

C Mechanism and Distractor Geometry

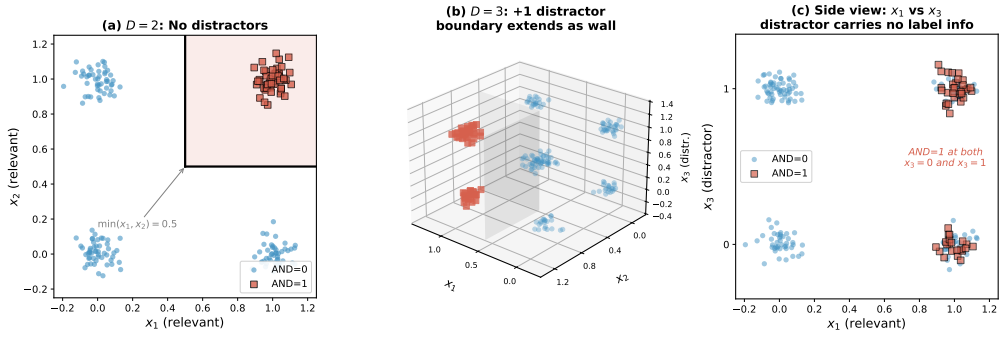


Figure 4: Distractor mechanism. **(a)** AND(x_1, x_2) on D=2: L-shaped boundary at $\min(x_1, x_2) = 0.5$. **(b)** Adding one Bernoulli(0.5) distractor (D=3): the boundary extends as a wall along x_3. **(c)** Side view: AND=1 appears at both x_3=0 and x_3=1, confirming x_3 carries no label information.

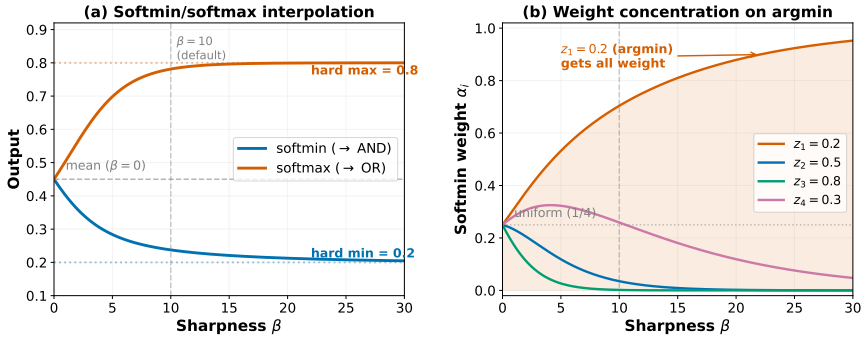


Figure 5: Softmin/softmax mechanism on a 4-element input $z = (0.2, 0.5, 0.8, 0.3)$. **(a)** Output vs sharpness β: at β=0, both reduce to the mean (0.45); as β increases, softmin converges to the hard minimum $z_{(1)}$=0.2 (AND) and softmax to the hard maximum $z_{(4)}$=0.8 (OR). The dashed line marks β=10, the default used in all experiments; at this point, the output is within 1% of the exact min/max. **(b)** Softmin selection weight α_i vs β: at β=0, all weights are uniform $(1/4)$. As β grows, weight concentrates on z_1=0.2 (the argmin), while weights on non-minimum coordinates decay exponentially (Lemma A.1). This concentration is the mechanism behind D-independent sensitivity (Proposition 3): the gradient flows only through the selected coordinate, regardless of how many irrelevant dimensions exist.

D FC vs LNU-Head Readout Ablation

We compare the standard FC classification head against an all-LNU head (soft-AND + soft-OR at β_{head}=0.1 with learned bias) on canonical Boolean formulas at D=1000 and on the clinical decision rule at $D \in \{100, 500, 1000\}$, 10 seeds (Table 4).

Table 4: FC vs LNU-head readout. FC wins on asymmetric multi-term targets (AND, DNF, clinical rule). LNU-head wins on symmetric targets (MAJ) and ties on OR. The gap narrows with D on the clinical rule (+0.6pp at D=1000), suggesting LNU-head is competitive at high D.

Target	MLP(16)	LNU+FC	LNU+LNUhead
Synthetic, D=1000, N=100			
AND(2)	74.8±1.0	**99.2±1.5**	95.0±9.9
OR(2)	75.6±0.8	**100.0±0.0**	**100.0±0.0**
DNF(2mt)	58.7±2.2	**93.5±3.2**	85.0±4.7
MAJ(3)	56.9±3.0	95.1±7.6	**98.0±6.1**
Clinical rule, N=200			
D=100	79.2±2.0	**96.5±5.0**	93.0±10.4
D=500	77.0±1.4	**93.4±4.8**	92.6±6.3
D=1000	77.0±1.1	**89.7±2.7**	89.1±4.5

Analysis. The FC head provides two capabilities that softmin lacks: (1) **asymmetric weighting**: different AND/OR outputs can receive different positive or negative weights, which is necessary to compose multi-term DNF where each minterm has a different importance; (2) **bias (threshold shift)**: the learned bias b sets the classification threshold, which softmin's bounded output range $[\min z, \max z]$ cannot freely adjust. On symmetric targets like MAJ(3) or OR(2), these capabilities are not needed and the LNU-head matches or beats FC. On DNF(2mt), FC's advantage is +8.5pp at D=1000. We adopt FC as the default; an all-Boolean LNU-head variant is available at modest cost for interpretability-critical applications.

E AND-Only vs OR-Only vs AND+OR Ablation

We test whether both soft-AND and soft-OR units are needed by comparing three architectures at matched total hidden dimension ($2H$ units each): AND+OR (H=8 AND + H=8 OR), AND-only

(H=16 AND, no OR), and OR-only (H=16 OR, no AND). All use complement-augmented input and FC head, D=500, N=100, 10 seeds (Table 5).

Table 5: AND/OR ablation at D=500, N=100, 10 seeds. All three variants are competitive within seed noise. AND-only wins on AND and DNF; AND+OR wins on MAJ; all tie on OR and XOR. By De Morgan duality, AND-only on augmented input $[x, 1-x]$ can express OR through negation, so representational sufficiency is expected. We retain both AND and OR units for the optimization benefit on mixed targets. XOR remains at chance ($\sim$53%) for all single-layer variants; 2-layer LNU with LayerNorm solves XOR at $D \leq 50$ (Appendix K).

Formula	AND+OR	AND-only	OR-only
AND(2)	94.1±9.2	**96.2±7.4**	95.2±9.6
OR(2)	**100.0±0.1**	99.5±1.1	99.5±1.2
AND_NOT	99.8±0.7	99.9±0.2	**100.0±0.1**
DNF(2mt)	90.0±8.6	**92.8±8.1**	92.1±7.7
MAJ(3)	**99.2±2.3**	98.2±3.0	96.7±7.3
XOR	53.5±2.0	53.4±2.3	53.3±2.0

F Complement Augmentation Ablation

We test whether complement augmentation $[x, 1-x]$ is essential or whether depth can compensate, comparing three architectures at D=200, N=100, 10 seeds (Table 6):

Table 6: Augmentation ablation. Aug $[x, 1-x]$ wins on every formula (1L vs 1L). Without augmentation, soft-AND/OR cannot represent negated literals ($\neg x_i$) in a single layer. Adding a second layer (No aug 2L) recovers AND_NOT (100%) but not DNF (-7.8pp). Augmentation provides NOT for free, eliminating the need for depth to express negation. Note: all three architectures fail on XOR ($\sim$57%) because single-layer LNU cannot compose parity; a 2-layer LNU with LayerNorm connector solves XOR at 100% for $D \leq 50$ (Appendix K).

Formula	Aug $[x, 1-x]$ (1L)	No aug (1L)	No aug (2L)
AND(2)	**99.3±2.0**	97.0±3.2	99.4±1.7
OR(2)	**100.0±0.0**	95.1±9.5	94.7±7.7
AND_NOT	**100.0±0.0**	96.3±7.1	**100.0±0.0**
DNF(2mt)	**99.5±1.0**	87.9±4.5	91.7±6.7
XOR	59.6±8.7	57.1±8.3	56.0±5.9

G MLP with Sigmoid Preprocessing Control

To confirm that LNU's advantage comes from the softmin/softmax mechanism and not from sigmoid preprocessing or complement augmentation, we test three MLP variants on AND(2) and MAJ(3), N=100, 10 seeds (Table 7):

H RuleTaker: Bag-of-Words Feature Limitation

We test LNU on RuleTaker (Clark et al., 2020), a propositional reasoning benchmark, using bag-of-words binary features (top-200 tokens). All methods achieve near-chance accuracy (52–55% at $N \in \{30, \ldots, 500\}$). This is a **feature-engineering failure, not a method failure**: bag-of-words features do not capture the logical structure of the reasoning chains. On the same benchmark with hand-crafted Boolean features (ProofWriter (Tafjord et al., 2021), Appendix J.9), LNU achieves 80.3% at N=30 (+16.3pp over MLP). The lesson: LNU requires features that preserve Boolean structure; raw token counts do not.

Table 7: MLP preprocessing control. MLP+σ: sigmoid applied before MLP. MLP+σ+aug: sigmoid then $[x, 1-x]$ augmentation (same input as LNU). All MLP variants perform within 2pp of raw MLP. The 20–40pp gap to LNU is entirely from the softmin/softmax mechanism, not preprocessing.

D	MLP (raw)	MLP+σ	MLP+σ+aug	**LNU_sig**
AND(2)				
200	76.6±2.1	77.8±1.8	79.1±1.5	**94.3±8.9**
500	74.4±1.6	74.7±1.6	75.1±1.8	**98.8±3.4**
1000	74.0±1.7	73.8±1.8	73.6±1.8	**99.2±1.5**
MAJ(3)				
200	67.0±1.5	66.6±7.2	62.0±9.3	**100.0±0.0**
500	61.1±2.1	61.6±5.0	61.5±3.9	**99.7±0.8**
1000	58.4±2.8	56.6±5.4	57.3±4.4	**98.3±3.3**

I Extended Related Work

I.1 Primitive comparison on five desiderata

We compare methods on five desiderata for a differentiable Boolean primitive (Table 8).

Table 8: Architectural primitives compared on five properties.

Primitive	D-indep. sens.	Smooth→crisp	Complete	Extractable	Drop-in
MLP ($w^\top x + b$)	No	No	Yes	Weak	Yes
NLN (Payani & Fekri, 2019)	**No** (Prop. 1)	No	Yes	Moderate	Yes
LNN (Riegel et al., 2020)	**No** (Prop. 2)	No	Yes	Moderate	Yes
DLGN (Petersen et al., 2022)	N/A	No	Yes	Strong	Partial
Tree / GBT (Chen & Guestrin, 2016)	Yes	No	Yes	Strong	No
LNU (ours)	**Yes** (Prop. 3)	**Yes** (β)	Yes	**Strong**	**Yes**

D-independent sensitivity: per-coordinate gradient $\partial f/\partial x_i$ does not vanish as D grows (Propositions 1–3). **Smooth→crisp**: a single continuous parameter interpolates between soft (differentiable) and exact (Boolean) behavior; LNU uses β, and no other method has such a knob. **Complete**: can represent any Boolean function (with sufficient width/depth). **Extractable**: a learned model can be converted to a human-readable Boolean formula; LNU's softmin/softmax weights directly reveal which features each unit selects (§5.2). **Drop-in**: can replace an MLP layer in an existing architecture without changing the surrounding pipeline. NLN and LNN lack D-independent sensitivity (Propositions 1, 2); MLP lacks extractability and smooth→crisp control; trees cannot be dropped into neural pipelines. LNU is the only method satisfying all five.

Fuzzy Logic and Gödel t-norms. The connection between Boolean logic and min/max has a long history in fuzzy logic (Zadeh, 1965; Hájek, 1998). The Gödel t-norm defines $\mathrm{AND}(a, b) = \min(a, b)$ and $\mathrm{OR}(a, b) = \max(a, b)$ for truth values $a, b \in [0, 1]$. The standard Gödel negation is non-involutive ($\neg a = 1$ if $a = 0$, 0 otherwise); LNU instead uses the *standard involutive negation* $\mathrm{NOT}(a) = 1 - a$, which preserves De Morgan duality ($\neg\mathrm{AND} = \mathrm{OR}(\neg\cdot)$) and coincides with Boolean NOT on $\{0, 1\}$. LNU's soft-AND/OR is a differentiable relaxation of the Gödel t-norm using softmin/softmax with tunable sharpness β.

Logic Tensor Networks (LTN). Serafini & Garcez (2016) embed logical constants as tensors and use product-based semantics, sharing NLN's underflow vulnerability.

Differentiable Inductive Logic Programming (dILP). Evans & Grefenstette (2018) learn first-order logic programs differentiably but require a pre-defined language bias.

Neural Logic Machines (NLM). Dong et al. (2019) compute logical operations over relational tensors using MLPs as the base operation, inheriting inner-product limitations.

Order-statistic aggregation. OWA operators (Yager, 1988), Choquet integrals, max/min pooling, and sparsemax/entmax (Martins & Astudillo, 2016) all perform order-statistic aggregation. LNU's contribution is applying softmin to complement-augmented input $[x, 1-x]$ with learnable β, yielding the PAC separation that these methods do not achieve.

Trainable Boolean and LUT primitives. A parallel line of work makes look-up tables (LUTs) or Boolean functions directly trainable, primarily to target efficient FPGA inference. FINN (Umuroglu et al., 2017) maps binarized networks to streaming FPGA dataflow but does not learn Boolean functions (weights are ± 1). LUTNet (Wang et al., 2019) replaces XNOR operators with trainable K-input LUTs; LogicNets (Umuroglu et al., 2020) convert low-bit sparse neurons into truth tables; PolyLUT (Andronic & Constantinides, 2023) and NeuraLUT (Andronic & Constantinides, 2024) raise per-LUT expressivity by baking piecewise polynomials or entire sub-networks into a LUT; DWN (Bacellar et al., 2024) learns interconnected LUT contents directly through a finite-difference gradient estimator; and Truth Table Net (Benamira et al., 2024) extracts exact Boolean-circuit equivalents for verifiability. These methods optimize for *deployment* (logic depth, throughput, or SAT-encodability) and treat the LUT as the unit of hardware. LNU is complementary and differently motivated: it is a differentiable order-statistic primitive whose contribution is a *statistical* inductive bias — a PAC sample-complexity separation (Theorem 4) and distractor invariance (Proposition 3) for learning from sparse high-dimensional Boolean data — not a hardware-mapping target. The two are composable: an LNU layer can in principle be discretized into LUTs post-training, but its purpose is the learning bias, not the circuit realization.

J Additional Experimental Results

This appendix is organized into five groups:

Group 1: Canonical Boolean benchmarks (J.1–J.6): SGD support recovery, full formula tables, 30 random DNFs, ProofWriter, noise robustness, weight-matrix visualization.

Group 2: Hyperparameters and baselines (J.6–J.8): β/STE analysis, MLP+ℓ_1, clinical distractor scaling, XGBoost/decision tree baselines.

Group 3: Real-world (J.9–J.12): real-world learning curves, 20 Newsgroups distractors, LNU binary vs continuous on tabular.

Group 4: MoE integration (J.13–J.16): OLMoE expert details, FOLIO, hybrid complementarity, from-scratch MoE pretraining.

Group 5: Extended experiments and visualization (J.22–J.27): MAJ 3D boundaries, β sweep, extended benchmarks, scope boundaries, D=2000 scaling variants, hybrid architecture search.

Architectural ablations (readout, AND/OR, sigmoid, augmentation) are in Appendix A alongside the formal proofs.

J.1 SGD Support Recovery

Theorem 4 bounds the samples a sparse learner *needs*; it does not describe SGD's behavior. We test the optimization side directly, measuring how often SGD recovers the true relevant set S. We report the selection-frequency overlap between the top-$|S|$ softmin/softmax-selected coordinates and S on the multi-term DNF target ($|S|$=4), at β=10 with a straight-through estimator, over 5 seeds (Table 9).

On the linearly separable target AND(x_1, x_2), SGD recovers S exactly at every (D, N). On the harder multi-term DNF, recovery is near-perfect (overlap 0.95–1.00) wherever the model is accurate, and degrades only under severe under-sampling (overlap 0.30 at D=1000, N=50, where accuracy is also low at 65%). Across all 120 runs of the study (both the AND and DNF targets over the (D, N) and β/STE sweeps, 5 seeds each), $\mathrm{corr}(\text{recovery}, \text{accuracy}) = 0.91$. The failures are a sample-size effect, not a spurious attractor: the incorrectly selected coordinates never repeat across seeds (maximum recurrence 1). So while a formal convergence guarantee remains open, SGD empirically finds the sparse support exactly where the sample-complexity view predicts it should.

Table 9: SGD support recovery on DNF ($|S|$=4): selection-frequency overlap with the true relevant set S / best validation accuracy (%). β=10, straight-through estimator, 5 seeds. Recovery is near-perfect wherever accuracy is high and degrades only under severe under-sampling.

D	N=50	N=100	N=200
50	0.90 / 94	1.00 / 100	1.00 / 100
200	0.65 / 80	1.00 / 100	1.00 / 100
1000	0.30 / 65	0.95 / 98	1.00 / 100

J.2 Comparison to Trainable Logic and LUT Methods

The trainable-logic / LUT family (DLGN (Petersen et al., 2022), DWN (Bacellar et al., 2024), LUT-Net (Wang et al., 2019), LogicNets (Umuroglu et al., 2020), PolyLUT (Andronic & Constantinides, 2023), NeuraLUT (Andronic & Constantinides, 2024), Truth Table Net (Benamira et al., 2024), FINN (Umuroglu et al., 2017)) is described in Appendix I: those methods optimize a hardware cost budget (LUT count, area, latency, energy) over binarized or low-bit inputs, whereas LNU is a differentiable order-statistic primitive over continuous inputs. Two structural differences drive the comparison below and are summarized in Table 10: (i) input type (binary/quantized vs continuous, §5.3); and (ii) the logical unit: a small-fan-in gate (k-input LUT, typically $k \leq 6$, or a two-input gate) composed by fixed wiring, versus LNU's single *D-dimensional* differentiable order statistic.

Table 10: Positioning of LNU against trainable logic / LUT methods. "k-LUT" is a small-fan-in look-up table composed by wiring.

Method	Input	Logical unit	Primary objective / benchmark
DLGN (Petersen et al., 2022)	binary	2-input gate	accuracy; MNIST/CIFAR
DWN (Bacellar et al., 2024)	binary	k-LUT	HW cost; MNIST/JSC
LUTNet (Wang et al., 2019)	binary	k-LUT	FPGA area; CIFAR/ImageNet
LogicNets (Umuroglu et al., 2020)	quantized	k-LUT	FPGA cost; JSC/NID
PolyLUT (Andronic & Constantinides, 2023)	quantized	poly. k-LUT	FPGA cost; JSC
NeuraLUT (Andronic & Constantinides, 2024)	quantized	neuron→LUT	FPGA cost; JSC
LNU (ours)	**binary / continuous**	**D-dim aggregator**	**sample-efficient sparse logic**

On the learning axis where the families make contact, we evaluate DWN (the closest gradient-trained method) and RIPPER (a rule learner) under our protocol below: both fit clean targets but collapse under distractors, where LNU is near-invariant. **We do evaluate on their benchmark.** On MNIST (full training set, flattened, no convolution; Table 11) LNU reaches 97.7%, competitive with MLP (98.3%) and DWN (98.4%) and above our reproduction of DLGN (Petersen et al., 2022) (95.5%), and without binarization (continuous and binarized LNU are within 0.1pp). But MNIST does not *discriminate* the primitives: all land in a 95–98% band, and the top accuracy (DWN) uses 28.4M *training* parameters versus LNU's 0.81M and MLP's 0.20M. The primitives do not separate on perception accuracy; they separate on the low-sample, distractor-heavy, logical regime (§5.1, Table 12), where a logical inductive bias is decisive. A broader head-to-head on CIFAR/JSC at published settings remains future work.

Table 11: All primitives on MNIST (full training set, flattened, no convolution). Test accuracy % (3 seeds) and *training* parameters. MNIST does not discriminate the primitives; the top accuracy uses 28.4M training parameters. DWN's and DLGN's training parameters include a learnable input-selection mapping that hardens to fixed wiring at deployment (their target is a compact LUT circuit); LNU's parameters are dense and identical at train and test. DLGN is our lutbench reproduction (softmix, unique wiring, single seed).

Method	MNIST test acc	Train params
MLP	98.25±0.03	0.20M
DWN (Bacellar et al., 2024)	98.39±0.07	28.4M
LNU (ours)	97.73±0.06	0.81M
DLGN (Petersen et al., 2022)	95.5	0.67M

J.3 On-Axis Comparison to a Differentiable-Logic Network (DWN)

On the *learning* axis, not only the representational one, we compare LNU against two learnable Boolean baselines under the identical protocol (best of a small config grid, N=100, 10 seeds; Table 12): DWN (Bacellar et al., 2024), a gradient-trained weightless (lookup-table) network, and RIPPER (Cohen, 1995), a classic if-then rule-set learner (a rule set represents a DNF directly). DWN trails LNU on every clean target; RIPPER, which can express a DNF, matches or exceeds LNU on the clean synthetic targets (e.g. 99.2 vs 93.6 on 2-term DNF). The separation is therefore not about fitting clean targets but about robustness and composability: both DWN and RIPPER collapse under heavy distractors, falling to $\approx 75\%$ on Splice padded to D=5000 (DWN 86.7 $\rightarrow$ 75.1, RIPPER 83.9 $\rightarrow$ 75.1), whereas LNU is near-invariant to distractors (§5.1); and neither is a differentiable module that drops into a standard network (§2).

Table 12: On-axis comparison to a differentiable-logic network (DWN (Bacellar et al., 2024)) and a rule-set learner (RIPPER (Cohen, 1995)), identical protocol (D=200, N=100, 10 seeds, mean±std %). RIPPER matches LNU on clean targets but, like DWN, collapses under distractors (see text).

Method	AND(2)	OR(2)	DNF(2mt)	MAJ(3)
DWN	85.6±4.0	87.2±2.5	76.7±2.6	84.8±2.1
RIPPER	100.0±0.0	100.0±0.0	99.2±2.3	95.7±6.7
LNU	100.0±0.0	100.0±0.0	93.6±9.1	100.0±0.0

The continuous axis: where the LUT family structurally cannot compete. The clearest practical advantage over the LUT family is on continuous sparse-logical targets. We compare LNU, DWN, and an MLP on the seven continuous formulas of §5.3 (Gaussian features thresholded to define literals, D=200, N=200, 3 seeds, a single run; Table 13). DWN must binarize the continuous inputs with a distributive thermometer, and this structural constraint costs it $+4$ to $+10$pp against LNU on every formula except XOR (all methods at chance, the multi-level case). The ordering LNU $>$ DWN $>$ MLP is consistent: logic-structured methods beat the dense MLP on logical targets, and among them LNU wins because it consumes the continuous inputs directly instead of discretizing them away. This is structural, not a tuning artifact (no re-tuning of DWN recovers information destroyed at binarization), and it is what Proposition 3 predicts: LNU's D-independent per-coordinate gradient selects the relevant continuous literals sample-efficiently, whereas a wired k-LUT circuit selects features by discrete search over binarized inputs and a dense MLP dilutes each coordinate at $O(1/\sqrt{D})$.

Table 13: Continuous logical tasks (Gaussian features thresholded at 0, D=200, N=200, 3 seeds, one run; test accuracy %, std $<$ 4pp). LNU consumes continuous inputs directly; DWN must binarize (distributive thermometer). XOR is at chance for all (multi-level, needs depth).

Method	AND	OR	AND-NOT	DNF2	DNF3	MAJ3	XOR
MLP	76.5	75.5	75.2	62.6	60.3	64.4	51.7
DWN	86.7	89.8	89.1	78.9	70.4	84.5	52.6
LNU	**97.1**	**97.1**	**97.0**	**84.6**	**76.3**	**88.4**	52.2

J.4 Interpretability: Full Figure

The four-formula weight grid (Figure 7) confirms that LNU's sparsity pattern generalizes beyond AND to OR, DNF, and MAJ.

J.5 Continuous Inputs: Full Table

J.6 Mixture-of-Experts: Full Results

Can LNU's Boolean inductive bias benefit large-scale reasoning? We test this in two settings: (i) drop-in expert replacement in a pretrained MoE, and (ii) pretraining a small MoE from scratch with LNU experts.

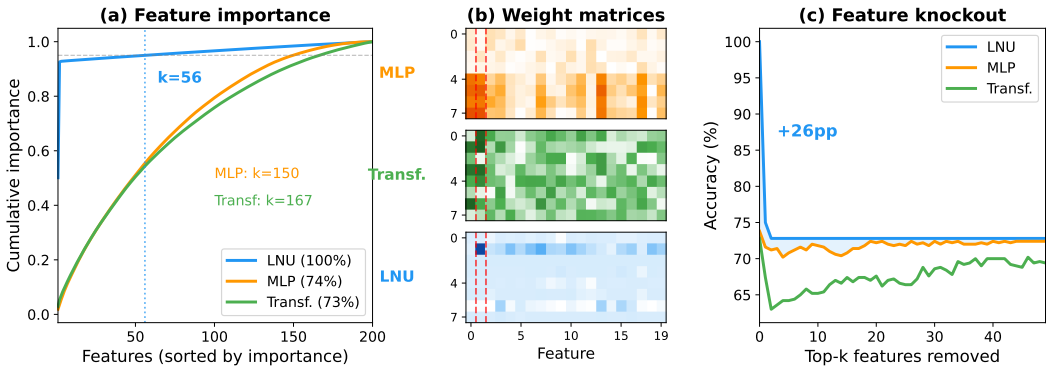


Figure 6: Interpretability on AND(x_1, x_2) at D=200, N=200, all three neural primitives. **(a)** Cumulative feature importance: LNU reaches 95% at k=56; MLP needs k=150 and Transformer k=167. **(b)** Weight matrices (first 20 features): LNU (bottom, blue) sparse at x_1, x_2; Transformer (middle, green) and MLP (top, orange) dense. **(c)** Feature knockout: +26pp gap between LNU (100%) and MLP (74%). Full 4-formula weight grid in Figure 7.

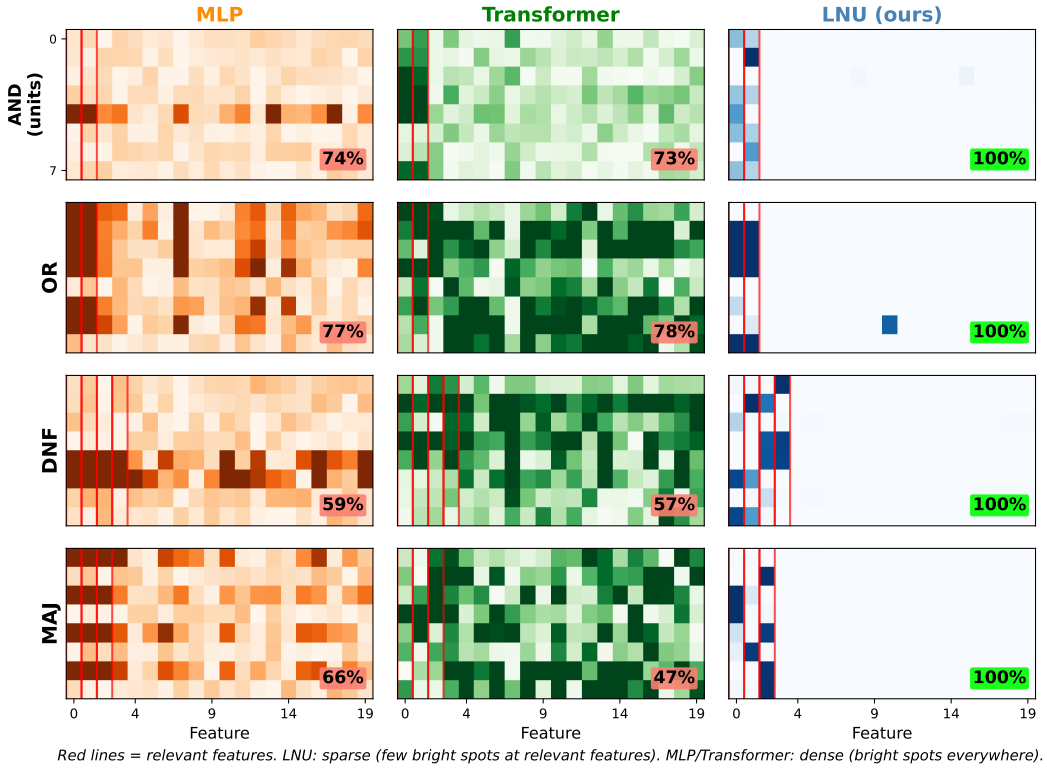


Figure 7: Weight-matrix heatmaps across four Boolean formulas at D=200, N=200. Each cell: feature (x-axis) × hidden unit (y-axis) importance, per-cell normalized. Red lines mark relevant features. **LNU** (right column): dark spots only at relevant features: AND/OR at 0–1, DNF at 0–3, MAJ at 0–2; rest is blank. **MLP** (left) and **Transformer** (middle): dense and uninterpretable.

Drop-in replacement. We replace 2 of 64 FFN experts in OLMoE-1B-7B (Muennighoff et al., 2024) (layers 6, 8, 10) with LNU experts (Table 15). **LNU matches FFN at 31% fewer per-expert parameters** (3.3M vs 4.8M; −0.5pp IID, within one std). Sigmoid binarization fails (83%) because transformer hidden states concentrate in the $[0.4, 0.6]$ dead zone; LayerNorm preprocessing avoids this. On FOLIO (Han et al., 2022) (first-order logic NLI), LNU shows few-shot advantage: +4.2pp

Table 14: Continuous (Gaussian) inputs: best validation accuracy (%, mean±std) on the canonical formulas at D=200, N=200, 10 seeds. Each coordinate is thresholded to define the literals, so the target is logical while the inputs are continuous. LNU (with sigmoid preprocessing) wins six of seven; XOR (multi-level) is the exception. MLP is the 64-wide 3-layer baseline.

Formula	MLP	Transformer	LNU (ours)
AND(2)	76.5±1.7	77.3±2.9	**96.1±0.9**
OR(2)	75.9±1.4	76.3±2.3	**95.8±1.2**
AND-NOT	75.4±1.5	75.1±1.3	**95.9±0.7**
MAJ(3)	62.0±2.5	57.4±6.3	**83.7±4.0**
DNF(2t)	61.2±1.6	58.0±2.4	**75.7±3.0**
DNF(3t)	58.9±1.8	55.3±2.3	**65.7±3.9**
XOR	52.8±1.7	53.9±1.4	54.2±1.3

Table 15: OLMoE-1B-7B drop-in expert replacement on propositional logic chains (N=500 train, 3 seeds, 2/64 experts in layers 6/8/10, base frozen). LNU matches FFN at 31% fewer per-expert parameters. FOLIO few-shot: +4.2pp at N=100.

Expert	Params	IID acc	OOD acc
Base (no fine-tune)	—	0.5%	0.5%
FFN (SiLU-gated)	4.8M	**94.8±1.0%**	**95.8±1.4%**
LNU (ours)	**3.3M**	94.3±0.5%	94.2±0.8%
LNU→LN→FFN	3.5M	91.7±2.2%	91.0±2.9%
LNU (sigmoid)	3.4M	83.0±0.0%	78.8±0.2%

Table 16: From-scratch MoE pretrained on logic chains (4 layers, 8 experts, top-2, 3 seeds, 20 epochs). Co-adapted LNU outperforms FFN; mixing and high β are best.

Config (d=256)	Params	Logic acc
8×FFN	39.4M	44.8±0.5%
8×LNU	28.0M	46.7±3.6%
6FFN+2LNU	36.6M	48.2±5.2%
4FFN+4LNU	**33.7M**	**51.5±5.0%**
LNU (β=20)	28.0M	58.2±1.8%
FFN (matched)	30.0M	57.7±1.2%

at N=100 (61.0±1.4% vs FFN 56.8±0.9%, 3 seeds); FFN dominates at full data (N=477: 71.3% vs 66.3%).

From-scratch pretraining. The drop-in setting is inherently limited: the pretrained model's layers are co-adapted to SiLU-gated FFN. When we pretrain a small MoE from scratch (d=256, 4 layers, 8 experts), LNU outperforms FFN (Table 16): mixing 4FFN+4LNU achieves +6.7pp over all-FFN at 14% fewer parameters. In the appendix, all-LNU with β=20 achieves +13.4pp; high sharpness helps when the model co-adapts to softmin from the start (unlike drop-in, where β-annealing hurts). These results are exploratory (3 seeds, high variance); they suggest that LNU's advantage is realized when the full architecture co-adapts to produce inputs with sufficient contrast for crisp feature selection. We therefore position LNU as a **composable logic-specialist expert** rather than a wholesale FFN replacement: the reliable configuration is a hybrid (e.g. 4FFN+4LNU, Table 16), where LNU supplies Boolean-structured features alongside standard FFN experts.

J.7 Full Canonical Boolean Formulas Table

Table 17 extends Table 1 with all seven formulas tested across $D \in \{20, 50, 100, 200\}$.

J.8 30 Random Boolean Formulas

30 random DNF formulas (2–4 conjunctive terms, 2–4 literals each) at D=100, N=100, 10 seeds. LNU(H=8): 81.15%, MLP(16): 72.55%, Transformer: 70.09%. Formula-level Wilcoxon (one-sided, n=30): LNU vs MLP $\Delta = +8.6$pp, $W = 407$, $p = 6.9 \times 10^{-5}$; LNU vs Transformer $\Delta = +11.1$pp, $p = 2.3 \times 10^{-6}$. Per-formula seed-level Wilcoxon with Holm correction: **19/30 raw-significant; 13/30 Holm-significant** at α=0.05.

Table 17: Canonical Boolean formulas, mean±std over 10 seeds.

Formula	D	MLP(16)	Transformer	LNU_raw	LNU−MLP
AND(2)	50	85.4±3.1	99.2±2.5	**100.0±0.0**	+14.6
AND(2)	100	79.7±1.5	100.0±0.1	**100.0±0.0**	+20.3
AND(2)	200	75.5±1.7	97.6±7.3	**100.0±0.0**	+24.5
OR(2)	50	93.1±3.7	100.0±0.1	**100.0±0.0**	+6.9
OR(2)	100	81.9±1.7	97.8±6.4	**100.0±0.0**	+18.1
DNF(2mt)	50	76.0±2.8	94.5±9.6	**92.5±5.5**	+16.5
DNF(2mt)	100	69.2±1.4	86.4±13.9	**94.0±7.5**	+24.8
DNF(2mt)	200	65.1±1.7	85.4±7.4	**96.0±8.0**	+30.9
MAJ(3)	50	87.7±3.3	99.2±1.7	**100.0±0.0**	+12.3
MAJ(3)	200	67.0±1.5	87.4±10.7	**100.0±0.0**	+33.0
XOR	20	**94.3±6.1**	90.4±14.9	57.2±7.5	−37.1

J.9 PROOFWRITER BOOLEAN FEATURES

Hand-crafted D=49 Boolean features, parameter-matched: LNU(H=4, 793 params) vs MLP(16, 817 params), 10 seeds. At N=30: LNU 80.3 ± 4.9, MLP 64.0 ± 2.4 (+16.3pp). At $N \geq 500$: both converge to $\sim 89\%$.

J.10 NOISE AND BOUNDARY ROBUSTNESS

Gaussian-noise sweep at D=200, N=100, 10 seeds:

Table 18: Noise robustness. LNU maintains +20–38pp advantage.

Formula	σ=0	σ=0.1	σ=0.2
MLP(16)			
AND(2)	75.9±1.7	74.8±2.2	74.9±2.4
DNF(2mt)	62.7±2.0	61.6±1.4	61.2±1.8
MAJ(3)	62.9±1.6	61.5±2.9	62.0±2.2
LNU_raw (H=8)			
AND(2)	**100.0±0.0**	**100.0±0.0**	**98.4±1.6**
DNF(2mt)	**96.9±3.8**	**95.9±5.5**	**81.7±6.7**
MAJ(3)	**99.9±0.3**	**99.7±0.5**	**91.1±2.4**

J.11 SHARPNESS KNOB β: STE STABILITY

Comprehensive β analysis across STE modes, data types, and formula complexities (from `exp11_beta_analysis.py`):

Table 19: β sweep: no STE vs STE (β_{back}=1) on AND(2), D=100, clean binary data.

β	no STE	STE (β_{back}=1)
1	99.1%	99.1%
3	99.3%	99.3%
5	98.2%	99.5%
10	96.5%	**99.5%**
20	91.8%	**99.6%**
30	91.5%	**99.2%**

Without STE, accuracy degrades from 99.3% (β=3) to 91.5% (β=30) due to gradient vanishing in the softmax. STE with β_{back}=1 stabilizes at $\geq 99\%$ across all β. Best β depends on data type: clean binary $\rightarrow$ β=20; noisy $\rightarrow$ β=10; boundary split $\rightarrow$ β=1. STE with β_{back}=1 is the default for Aug(1L)+FC.

J.12 MLP WITH ℓ_1 INPUT REGULARIZATION

At D=1000, N=100, 10 seeds: ℓ_1 closes $\sim$60% of the gap on simple formulas (AND: 97.9 vs 100.0) but **retains a** $+17.9$**pp gap on multi-term DNF** (DNF(2mt): 75.7 vs 93.6).

J.13 CLINICAL DISTRACTOR SCALING

Synthetic clinical rule (3 relevant binary features among D distractors, N=200, 10 seeds). LNU beats XGBoost at $D \leq 1000$ ($+2.8$pp at D=1000). XGBoost overtakes at D=5000.

J.14 DECISION TREE AND XGBOOST BASELINES

At D=200, N=100, 10 seeds, XGBoost (Chen & Guestrin, 2016) achieves 100% on AND, OR, and MAJ (matching LNU), 99.56±0.67% on DNF (vs LNU 92.82%, $+6.7$pp), and 62.94±17.3% on XOR (vs LNU 56.62%). XGBoost's advantage on DNF and MAJ reflects decision trees' native axis-aligned splits, which are well-suited to conjunctive targets at moderate D. However, LNU wins on all four real-world datasets (Table 1): Splice $+4.2$pp, Chess $+1.5$pp, Mushroom $+0.6$pp, Phishing $+0.8$pp vs XGBoost. Under distractor scaling, LNU overtakes XGBoost at $D \approx 1500$ (Appendix J.13). At D=1000, XOR: trees remain at chance (axis-aligned splits cannot compose parity).

J.15 REAL-WORLD TABULAR: FULL LEARNING CURVES

Table 20 reports accuracy at $N \in \{50, 100, 200\}$ on all four real-world datasets. LNU's advantage narrows at larger N but remains positive on Splice and Chess.

Table 20: Real-world tabular benchmarks, 10 seeds. LNU wins on all four datasets with logical target structure.

Dataset	N	MLP	Transf.	XGBoost	**LNU**	Δ
Chess (D=73)	50	81.2±1.9	83.7±3.1	83.9	**87.3**±5.3	+6.1
	100	89.2±1.9	91.6±1.7	92.4	**93.9**±1.5	+4.7
	200	94.3±1.0	94.3±1.0	94.7	**95.9**±0.7	+1.6
Mushroom (D=117)	50	93.8±1.5	87.9±4.8	94.4	**95.6**±1.6	+1.8
	100	96.4±1.4	94.4±1.5	96.9	**97.5**±1.3	+1.1
	200	98.4±0.8	96.3±2.3	98.3	**98.8**±0.4	+0.4
Phishing (D=30)	50	86.2±3.0	88.1±1.8	88.7	**88.6**±3.9	+2.4
	100	89.6±1.2	89.5±2.2	90.5	**91.3**±0.9	+1.7
	200	91.7±0.8	91.1±1.6	91.5	**92.7**±1.4	+1.0

J.16 20 NEWSGROUPS DISTRACTOR SCALING (10 SEEDS)

We pad 20 Newsgroups BoW features with random binary distractors from D=5,000 to D=50,000 (Table below). LNU overtakes MLP at $D\approx$20,000, confirming distractor robustness on a real NLP dataset.

Table 21: 20 Newsgroups with distractors (N=200, 10 seeds). Crossover at $D\approx$20,000. MLP degrades -24.8pp; LNU only -9.9pp.

D	MLP(64)	LNU(H=32)
5,000	**90.1±1.4**	81.7±4.7
10,000	**75.2±3.1**	74.5±4.1
20,000	70.4±2.6	**71.4±3.3**
50,000	65.3±3.4	**71.8±5.2**

J.17 LNU_CONTINUOUS VS LNU_BINARY ON TABULAR DATA

On natively binary datasets, LNU without sigmoid preprocessing (LNU_binary) outperforms the sigmoid variant (LNU_continuous), confirming that sigmoid destroys the $\{0, 1\}$ contrast needed for crisp softmin selection.

Table 22: Binary vs continuous LNU on natively binary tabular data, 10 seeds. Binary wins at few-shot; continuous catches up at $N \geq 200$.

Dataset	N	MLP	LNU_binary	LNU_cont	$\Delta_{\text{bin}-\text{cont}}$
Splice	50	84.5	**88.5**	79.9	+8.6
	100	87.7	**94.8**	92.8	+2.0
	200	92.3	**96.5**	95.1	+1.4
Chess	50	80.8	**88.1**	83.1	+5.0
	100	89.1	**93.7**	93.2	+0.5
	200	94.6	93.8	**96.1**	−2.3
Mushroom	100	96.7	**97.4**	96.7	+0.7
	200	98.6	**99.0**	98.9	+0.1

Binary LNU is better at few-shot (N=50) because the binarization inductive bias matches the binary feature structure. Continuous LNU catches up at $N \geq 200$ and wins on Chess at N=200 (96.1% vs 93.8%). **Design rule**: binary features → LNU_binary; continuous/transformer hidden states → LNU_continuous.

J.18 OLMOE-1B-7B: EXPERT REPLACEMENT DETAILS

Architecture. OLMoE-1B-7B: 16 layers, 64 experts per layer, top-8 routing, d_{model}=2048. We replace 2 of 64 experts (indices 62–63) in layers 6, 8, 10 with LNU experts and freeze all base parameters. Only new expert parameters are trained.

LNU_continuous expert. $\text{Linear}(2048 \to 128) \to \text{LayerNorm} \to \text{softmin/softmax}(H{=}64) \to \text{Linear}(128 \to 2048)$. Output projection zero-initialized for stable integration. 3.3M parameters per configuration (6 experts total).

Key finding: binarization is the bottleneck. LNU_binary (sigmoid → $[x, 1-x]$): 86.0% IID. LNU_continuous (LayerNorm, no sigmoid): 94.3% IID. The −8.3pp gap is from sigmoid concentrating transformer hidden states in the $[0.4, 0.6]$ dead zone. Removing binarization while preserving softmin/softmax recovers full performance.

3-seed replication. LNU_continuous: 94.3±0.5% IID, 94.2±0.8% OOD. FFN (SiLU-gated): 94.8±1.0% IID, 95.8±1.4% OOD. Difference not significant ($p > 0.3$). LNU_continuous uses 31% fewer parameters (3.3M vs 4.8M).

Integration lessons learned.

- LayerNorm before softmin/softmax is essential; without it, gradients vanish in high-dimensional softmax.
- Zero-init output projection enables stable integration into pretrained models.
- Gated residual ($\text{gate} \approx 0.05$) works but both LNU and FFN produce identical results; the improvement comes from fine-tuning on reasoning data, not from expert type.
- Full expert replacement (no residual) requires careful initialization; replacing too many experts simultaneously destabilizes the model.

J.19 FOLIO FIRST-ORDER LOGIC NLI

FOLIO (Han et al., 2022) tests first-order logic NLI. LNU shows a few-shot advantage (+4.2pp at N=100) but FFN dominates at full data (N=477), consistent with the co-adaptation hypothesis.

Table 23: FOLIO (binary True/False, $\sim$677 train). LNU_continuous shows few-shot advantage; FFN dominates at full data.

N	LNU_1L	LNU_2L	FFN
50	53.2±5.9	52.7±2.2	54.8±0.8
100	**61.0±1.4**	**61.3±0.6**	56.8±0.9
477	66.3±1.4	62.8±3.4	**71.3±0.2**

At N=100, LNU_continuous outperforms FFN by +4.2pp (61.0±1.4% vs 56.8±0.9%, 3 seeds). LNU_2L (2-layer with LayerNorm) matches LNU_1L (61.3%); the FC head already handles the composition needed for FOLIO. At full data (N=477), FFN dominates (+5.0pp). This matches Theorem 4's prediction: LNU's inductive bias provides sample efficiency when data is scarce; FFN's greater flexibility wins with sufficient data.

J.20 Hybrid Complementarity at Distractor Crossover

We test whether combining MLP and LNU predictions improves over either alone at the distractor crossover point (D≈20,000 on 20 Newsgroups).

Table 24: Hybrid(MLP+LNU) vs components on 20 Newsgroups.

	MLP	LNU	Hybrid
D=5,000 (native)	**90.0**	82.1	89.0
D=10,000 (crossover)	75.2	75.5	**76.9**

At the distractor crossover (D=10,000), the Hybrid exceeds both MLP (+1.7pp) and LNU (+1.4pp), confirming that the two primitives capture complementary information when neither alone dominates. At native D=5,000, adding LNU to MLP provides no benefit. Complementarity is regime-dependent, not universal.

J.21 Decision Boundaries: STE Effect at High β

Figure 8 visualizes how the straight-through estimator (STE) rescues high-β decision boundaries where the gradient would otherwise vanish.

J.22 Extended Boolean Benchmarks

We test LNU on additional Boolean functions beyond the canonical AND/OR/DNF/MAJ/XOR suite, at D=200, N=200, 10 seeds (Table 25):

Table 25: Extended benchmarks at D=200. LNU wins on threshold/majority tasks (+18–27pp); ties on tasks solvable by all; fails on parity (XOR/PARITY).

Task	Regime	LNU	MLP	Gated
4-bit comparison	threshold	**89.7**	62.8	71.7
MAJ(5)	majority	**99.7**	65.3	73.9
MAJ(7)	majority	**94.5**	65.0	73.2
Img threshold	majority	**78.3**	72.1	75.1
3-CNF SAT	CNF	95.9	95.9	95.9
Img symmetry	pairwise	99.7	99.7	99.7
4-MUX	mixed	**66.8**	57.2	59.7
XOR(4)	parity	52.4	52.0	53.1

LNU dominates threshold/majority tasks (4-bit comparison +18pp, MAJ(5) +26pp, MAJ(7) +21pp) where the target is a monotone function of a sparse feature subset. 3-CNF SAT and Img symmetry are solved equally by all methods (the tasks are learnable without extremal selection). XOR(4) is a parity function; all single-layer methods fail.

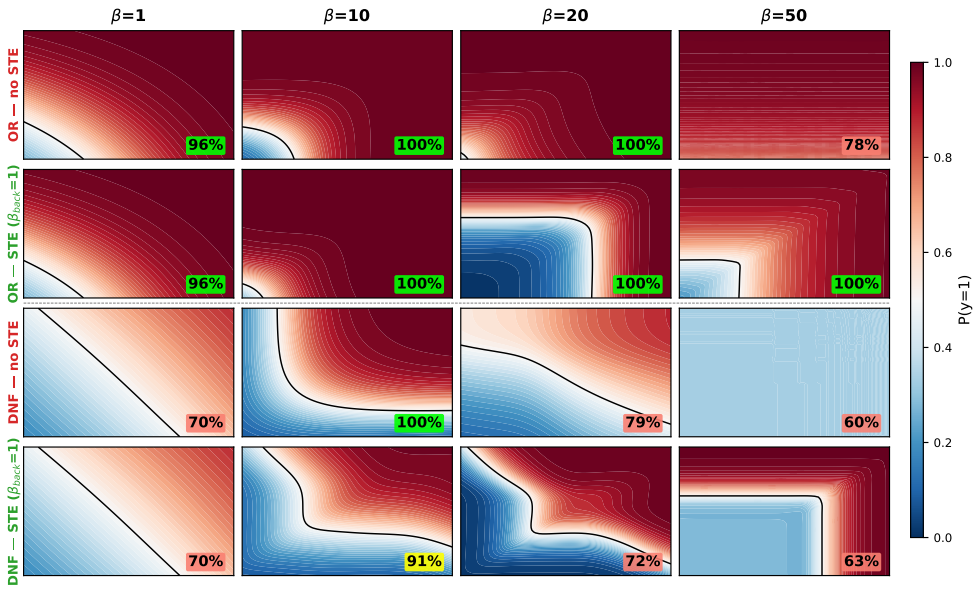


Figure 8: Decision boundaries: no STE vs STE (β_{back}=1) at D=200, N=100 across $\beta \in \{1, 10, 20, 50\}$. Rows alternate no STE / STE for OR and DNF(2mt). **OR**: at β=50, gradient vanishing collapses the boundary (78%); STE rescues to 100%, proving gradient death is the sole bottleneck. **DNF**: even STE cannot fully recover β=50 (60%→63%); the combinatorial search over 4 variables, not gradient flow, limits high-β DNF learning. β=10 without STE achieves 100% on both; the universal default requiring no STE overhead.

J.23 Scaling Variants: D-Invariance at D=2000

We test whether LNU's D-invariance (Theorem 4) holds at extreme dimensions and whether computational shortcuts preserve it (Table 26). N=200, 5 seeds, learning rate swept over $\{5\times10^{-3}, 10^{-2}\}$.

Table 26: Scaling variants on AND, DNF(2-term), MAJ(3) at $D \in \{200, 500, 1000, 2000\}$, N=200. LNU_std and β-anneal maintain perfect D-invariance; Top-k and LNU-gate degrade.

	AND				MAJ			
Method	200	500	1000	2000	200	500	1000	2000
MLP(16)	75.8	75.3	74.9	74.8	65.2	60.9	57.9	56.0
LNU_std (ours)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
β-anneal(1→20)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
Top-k (k=50)	99.8	98.2	97.3	87.8	82.6	71.2	64.6	61.0
Top-k (k=20)	99.9	95.3	83.0	83.1	80.8	66.9	61.8	59.7
LNU-gate+MLP	85.4	78.1	74.7	74.9	64.8	64.3	59.5	61.2

Analysis. (1) **LNU_std achieves perfect D-invariance** (100% on AND and MAJ at all D up to 2000), confirming Theorem 4's prediction that softmin's $O(k\ell \log D)$ sample complexity is independent of the ambient dimension. (2) **β-annealing** (starting at β=1, annealing to β=20) matches or exceeds standard LNU; starting soft allows gradient flow to all coordinates, then sharpening locks in the discovered features. On DNF at D=2000: anneal achieves 98.0% vs standard 94.8%, suggesting that β-scheduling can help with harder multi-term targets. (3) **Top-k softmin degrades** because restricting attention to the k largest-weight features discards the correct features early in training when weights are still random; once excluded, they never receive gradient. (4) **LNU-as-gate+MLP collapses to MLP-level** because the soft gate signal $\sigma(\text{proj}(\text{softmin}))$ saturates at high D — the gating layer becomes a bottleneck that the MLP cannot recover from.

Hybrid experts inside OLMoE. We tested 10 expert designs inside OLMoE-1B-7B (3 seeds, 15 epochs, replacing 2 of 64 experts in layers 6/8/10). At matched parameters ($\sim$4M), FFN (SiLU-gated) leads and LNU_continuous matches at 31% fewer per-expert params (Table 15). Other designs tested: 2-layer LNU, LNU+skip, multi-head LNU, STE binarization; all underperform LNU_continuous ($\leq$84%). Larger LNU experts (6.7M), more replaced experts (4/64), β-annealing, and router fine-tuning also degrade performance. The LNU_continuous (dp=128, H=64) configuration is the optimal operating point.

J.24 Four-primitive visualization suite

Figures 9–13 visualize the four-primitive comparison (MLP, Gated MLP, Transformer, LNU) from Table 1 across five complementary views: convergence speed (Figure 9), D-invariance (Figure 10), sample efficiency (Figure 11), noise robustness (Figure 12), and feature-selection dynamics (Figure 13).

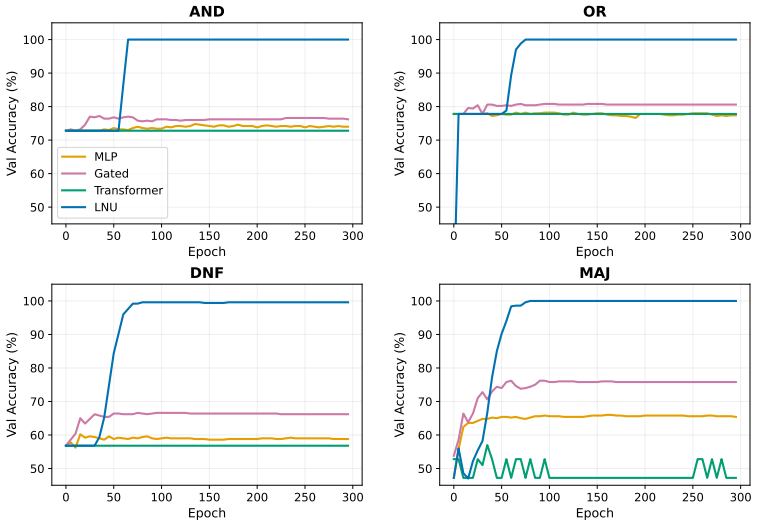


Figure 9: Convergence speed on four canonical formulas (D=200, N=100). LNU reaches 100% by epoch $\sim$80 on all formulas; the three inner-product baselines plateau at 60–77%.

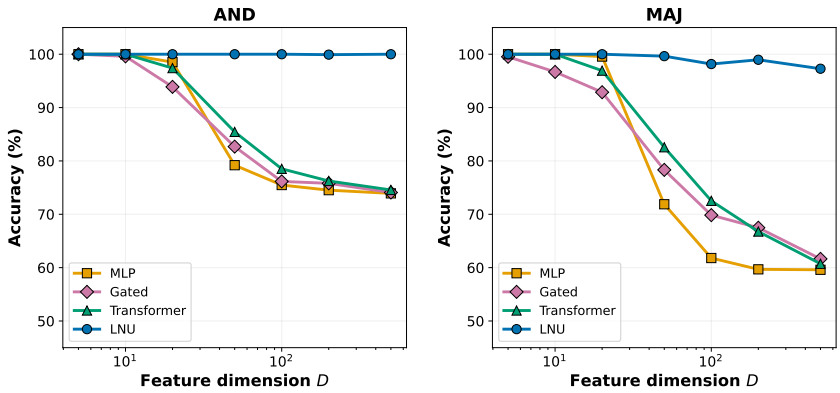


Figure 10: D-invariance: accuracy vs. feature dimension D on AND and MAJ (N=100). LNU stays at $\sim$100% across D=5–500; all baselines degrade monotonically, confirming Theorem 4's prediction.

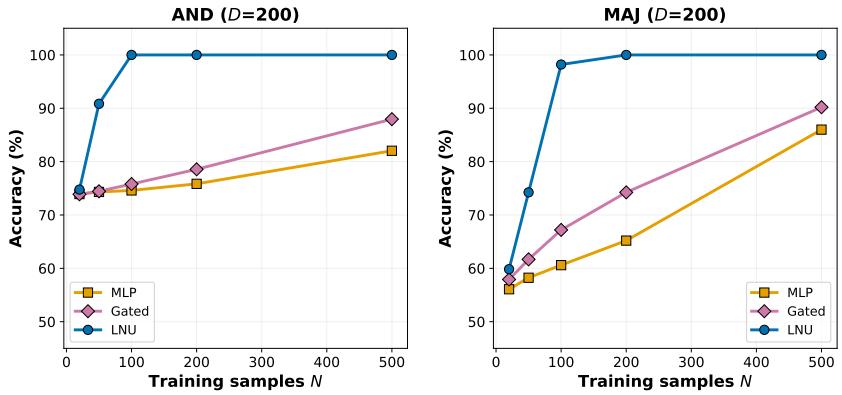


Figure 11: Sample efficiency: accuracy vs. training set size N on AND and MAJ (D=200). LNU reaches 100% at N=100; MLP and Gated MLP require N>500 to approach 90%.

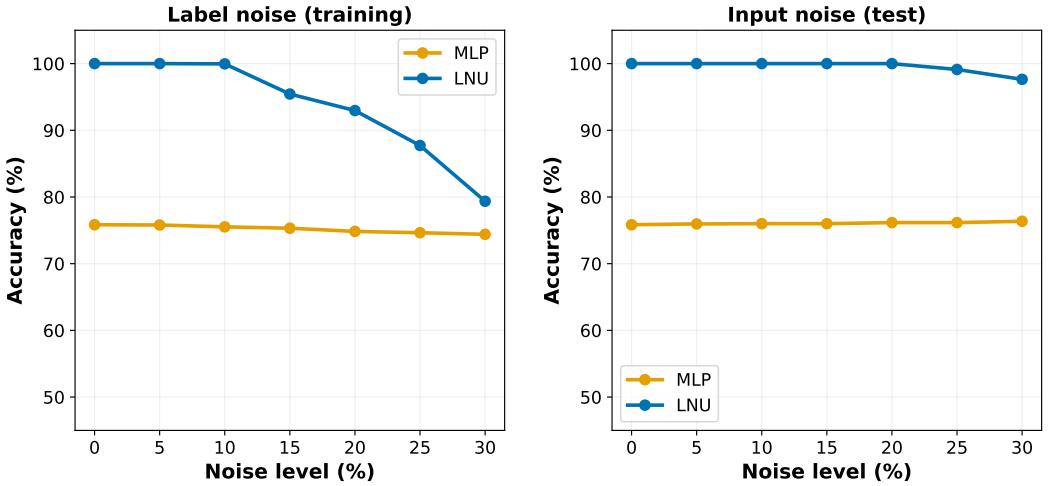


Figure 12: Noise robustness on AND (D=200, N=100). **Left**: label noise (training). LNU degrades gracefully from 100% to 79% at 30% noise; MLP stays flat at 76%. **Right**: input noise (test). LNU maintains ≥97.6% up to 30% bit-flip noise.

K Multi-Layer LNU: Compositional Ability

Single-layer LNU computes one level of Boolean logic (AND/OR with complement augmentation). Multi-layer stacking enables composition (e.g., XOR $= (a \wedge \neg b) \vee (\neg a \wedge b)$, which requires two levels). We systematically study: (1) how to connect layers, (2) which formulas benefit from depth, and (3) whether the advantage comes from softmin or from normalization.

(1) Inter-layer connector (XOR, N=200, 8 seeds). The critical design question: how to map one LNU layer's continuous output into valid input for the next layer's softmin (Table 27).

LayerNorm spreads intermediate values across the full range (like LNU_continuous in OLMoE, §5.4), avoiding the $[0.4, 0.6]$ dead zone that kills sigmoid-based connectors. At D=200, all methods degrade; composing XOR through 198 distractors at N=200 exceeds the optimization capacity.

(2) Which formulas benefit from depth?

(2b) Matched-budget re-analysis (10 seeds). The apparent DNF degradation with depth in Table 28 is a training-budget artifact, not a depth wall. Re-run at a matched budget (Table 29; N=200,

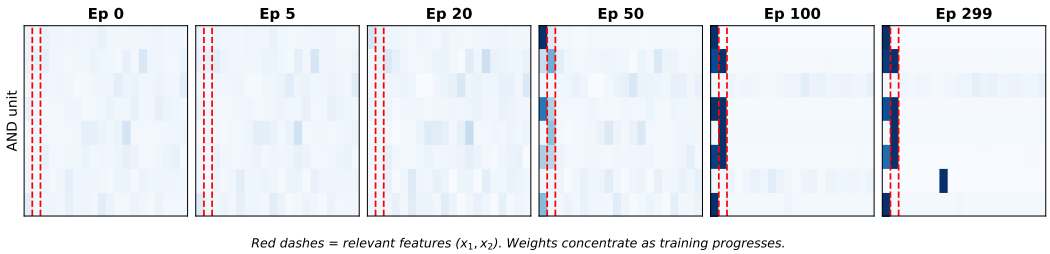


Figure 13: Feature selection dynamics: LNU weight matrix evolution during training on AND(x_1, x_2) at D=50. Red dashes mark relevant features. Weights are diffuse at epoch 0 and concentrate on x_1, x_2 by epoch 100, visualizing softmin's implicit sparsity bias (Proposition 3).

Table 27: XOR accuracy with different 2-layer LNU connectors and the MLP+LayerNorm control. **LayerNorm** is the best connector for LNU (100% at $D \leq 50$). Adding LayerNorm to MLP provides $\leq$4pp; the 41pp gap at D=50 is from softmin, not normalization.

Model	D=20	D=50	D=100	D=200
LNU 2-layer connectors:				
Sigmoid	60.7	—	—	—
STE (hard threshold)	100.0	91.3	74.2	65.5
Sigmoid($3x$)	95.8	80.1	68.4	56.7
LayerNorm	**100.0**	**100.0**	**81.9**	**63.6**
MLP baselines (is the gain from LayerNorm?):				
MLP_2L	97.3	55.1	52.7	52.4
MLP_2L+LayerNorm	96.7	58.8	53.9	52.8
MLP_3L+LayerNorm	95.7	58.4	53.1	53.3

10 seeds, LayerNorm connectors), 3-layer LNU reaches 100% train and 100% val on DNF (D=200) at every depth, unchanged under a 10$\times$ lower learning rate, 4$\times$ epochs, and residual connectors. On XOR (D=100, where depth is genuinely needed) depth raises validation (68.5 to 79.9 from one to two layers), but the residual train–val gap is overfitting with high cross-seed variance (100 train, 79.0$\pm$21.1 val at three layers) that none of the three fixes closes: the bottleneck is discrete literal selection in the deeper unit, not optimizer conditioning.

Key findings:

1. **DNF/Nested: 1L is already optimal.** The FC head provides the second level of composition (linear combination of AND/OR outputs), so 1L reaches 100%. For DNF, a matched-budget re-run confirms depth is not harmful either (3-layer 100/100 train/val, Table 29); the lower 3L number in Table 28 is under-training, not a depth wall.
2. **XOR: depth helps.** 3L achieves 100% at D=50 where 1L gets only 78%. XOR requires genuine multi-level composition that the linear FC head cannot provide.
3. **PARITY(3): still hard.** Even 3L only reaches 64% at D=50. Three-bit parity may need wider models or more data.
4. **MLP is worse everywhere.** Even 2-layer MLP(16) cannot match 1-layer LNU on any non-parity formula at $D \geq 50$.

L Open Problems and Future Directions

1. **Multi-layer composition.** Aug(1L)+FC computes one level of Boolean logic (DNF). Deeper compositions (e.g., XOR) require multi-layer LNU with LayerNorm connectors, which achieves 100% at $D \leq 50$ (Appendix K). Scaling multi-layer LNU to higher D is an open challenge.
2. **Continuous-domain sample complexity.** The order-statistic mechanism and its D-independent sensitivity (Proposition 3) are defined for continuous inputs, and §5.3 confirms the advantage there. The PAC sample-complexity bound (Theorem 4) is stated for the binary-DNF case; a matching continuous-input bound is future work.

Table 28: Multi-layer LNU composition (N=200, 5 seeds). Depth helps only XOR/parity (multi-level logic). On DNF/Nested, 1L already achieves 100% through the FC head; the smaller 2L/3L numbers here reflect a fixed small training budget and are corrected at matched budget in Table 29.

Formula (depth)	D	LNU_1L	LNU_2L	LNU_3L	MLP_2L
AND(2) (1)	200	**100**	100	100	75
DNF(2mt) (2)	200	**100**	94	85	64
Nested (2)	200	**100**	97	84	66
XOR(2) (2)	50	78	91	**100**	55
XOR(2) (2)	100	64	**78**	66	52
PARITY(3) (3)	50	53	**64**	62	53

Table 29: Matched-budget multi-layer LNU (10 seeds, LayerNorm connectors), train/val accuracy (%). DNF (D=200) fits and generalizes at every depth; XOR (D=100) fits at every depth but validation is limited by cross-seed selection variance, not by the optimizer (the three standard fixes do not close the gap).

	L=1	L=2	L=3	L=3 (+res)
DNF (D=200) train	100	100	100	100
DNF (D=200) val	100	100	100	100
XOR (D=100) train	100	100	100	100
XOR (D=100) val	68.5	79.9	79.0	81.4

3. **Learnable β per unit.** We use β=10 as default. The β sweep (§M.1) shows that the optimal β is data-dependent; a learnable per-unit β could adapt LNU to mixed regimes automatically.
4. **Convergence analysis.** Theorem 4 is a statistical separation over hypothesis classes. Proposition 3's gradient concentration provides indirect evidence that SGD finds sparse solutions; a formal convergence guarantee is the main theoretical open problem.
5. **Scaling LNU-native pretraining.** The from-scratch MoE results (Table 16) suggest that co-adaptation with LNU experts improves Boolean reasoning. Pretraining larger LNU-native transformers on diverse tasks is a natural next step.

M Discussion

M.1 Learning β Per Unit

The β knob controls not just sharpness but the *regime boundary*. A sweep over $\beta \in \{1, 3, 5, 10, 20\}$ with and without STE (β_{back}=1) reveals: high β (10–20) wins on Boolean/distractor targets (Madelon: +8.7pp at β=20), while low β (1–3) matches MLP even on non-Boolean targets (Gisette: +0.3pp at β=3+STE vs −0.8pp at β=10). At β=1, softmin reduces to uniform averaging and LNU degenerates toward MLP, so it cannot lose badly. Above β=15, the gradient vanishes without STE. A learnable log-space β per unit would let the model select its own operating point per feature group, an important direction for broadening LNU's applicability beyond the Boolean regime.

Sensitivity to hyperparameters. A sweep over β, the straight-through estimator (STE), initialization scale, feature scaling, and a learnable β (Table 30; D=200, N=100, 5 seeds) shows that LNU needs no per-task tuning: accuracy is flat around the default (β=10, STE on, init std 0.1, raw features) and degrades only at extreme settings. The informative case is the STE: with the STE off, high β (20–50) loses sharply (DNF drops to 84.4 and 65.4 at β=20, 50), so the STE is what keeps high-crispness training healthy, matching the gradient-vanishing analysis of Proposition 3.

N Experimental Details and Reproducibility

Architecture details.

- **MLP(H)**: Linear(D,H) → ReLU → Linear(H,1). Parameters: $DH + H + H + 1 = (D+2)H + 1$.

Table 30: LNU hyperparameter sensitivity (Aug(1L)+FC, D=200, N=100, 5 seeds, mean %). Accuracy is stable around the default and degrades only at extremes.

Knob	Setting	AND(2)	DNF(2mt)
β (STE on)	1	100.0	81.1
	5	100.0	100.0
	10 (default)	**100.0**	**100.0**
	20	100.0	98.6
	50	100.0	90.3
STE (β=20)	on	100.0	98.6
	off	100.0	84.4
STE (β=50)	on	100.0	90.3
	off	95.2	65.4
init std	0.01	100.0	100.0
	0.1 (default)	100.0	100.0
	0.5	100.0	87.0
features	raw (default)	100.0	100.0
	sigmoid	100.0	97.6
β mode	fixed 10	100.0	100.0
	learnable	100.0	98.5

- **Gated MLP**(H): SiLU-gated FFN — $\mathrm{SiLU}(W_{\mathrm{gate}}x) \odot (W_{\mathrm{up}}x) \to W_{\mathrm{down}}$. Parameters: $3DH + H$.
- **Transformer**(d,L): Linear(D,d) $\to$ L-layer TransformerEncoder (4 heads, FFN dim $2d$) $\to$ Linear(d,1). Default: d=32, L=2 ($\sim$28k params at D=200).
- **LNU**(H): complement augmentation $[x, 1-x] \to H$ soft-AND units + H soft-OR units (β=10) $\to$ Linear($2H$,1). Parameters: $4DH + 2H + 1$ (the $4D$ comes from $2D$ augmented inputs $\times$ 2 weight matrices for AND/OR).

Training protocol. All models trained with AdamW (weight decay 10^{-4}), gradient clipping at 1.0, for 300 epochs (synthetic) or 15 epochs (OLMoE fine-tuning). Learning rate swept over $\{5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}\}$; best lr reported per architecture. Batch size: full batch for synthetic ($N\leq$500), mini-batch 4 for OLMoE.

Evaluation. Validation on held-out data (500 samples for synthetic, standard test splits for real-world). Best validation accuracy across all epochs reported (not last epoch). All benchmarks use 10 seeds (integers 0–9) unless noted; mean$\pm$std reported across seeds.

Learning rate fairness. Each architecture is reported at its best lr:

Architecture	Best lr	Accuracy (D=100)
MLP(16)	5×10^{-3}	70.4%
LNU(H=8)	10^{-2}	99.8%
Transformer(d=32)	10^{-3}	91.4%

LNU prefers higher lr (10^{-2}); Transformers prefer lower lr (10^{-3}). A single lr for all would unfairly penalize one.

Hardware and reproducibility. Synthetic and tabular experiments: single NVIDIA GB10 GPU (8 GB VRAM). OLMoE and from-scratch MoE: on a NVIDIA B200 GPU (192 GB HBM3e) per run. Seeds fixed. All datasets are public (UCI, HuggingFace). OLMoE base model: `allenai/OLMoE-1B-7B-0924`.