LAWS: Learning from Actual Workloads Symbolically

A Self-Certifying Parametrized Cache Architecture
for Neural Inference, Robotics, and Edge Deployment

Gregory Magarshak
gmagarshak@faculty.ienyc.edu

Abstract

We introduce *Learning from Actual Workloads Symbolically* (LAWS), an inference-time architecture deployable above any trained neural network without modification. LAWS maintains a dynamically growing library of *parametrized experts*—cheap computational patterns, ranging from precompiled GPU kernels to tiny neural networks, created automatically from observed inference queries and certified formally for correctness. New queries are routed to the cheapest certified expert whose *validity domain* contains the query; the base model runs only on genuine cache misses.

The central technical contribution is the *self-certification theorem*: a trained network's weights encode, without any additional training or inference, a Lipschitz constant $\Lambda(W)$ certifying the validity of every expert. Routing radii are defined in the Probabilistic Language Trie (PLT) metric [15]; experts are correct on all inputs (Theorem 3), and routing radii determine which expert handles each query. No warmup, no proxy model, no retraining is required. LAWS strictly generalizes three prior approaches: (i) KV caching (degenerate case: zero validity radius, identity expert); (ii) Mixture of Experts (degenerate case: fixed library, no online creation); and (iii) manual symbolic AI (Cyc [14], Wolfram Alpha [22]), whose vocabularies require human authorship, whereas LAWS discovers its symbolic vocabulary automatically from the model's training distribution (Theorem 11).

The name reflects a deeper analogy. *Scientists discover natural laws from actual observations*—not by legislating them, but by identifying the invariant patterns in empirical data. LAWS does the same computationally: it discovers the "laws" governing a trained model's behavior—the regularities that hold across families of similar inputs—and encodes them as cheap certified experts. This mirrors how biological intelligence works: animals and humans learn by recognizing situations from experience and executing cached strategies, invoking deliberate computation only for genuinely novel inputs (Kahneman [11]). The model's pre-trained weights encode an innate prior library (Chomsky [7]); deployment experience calibrates and extends it.

We prove: (1) expert libraries grow at rate $O(2^{H(P_{\mathcal{M}})} \log N)$ under stationary distributions (equivalently $O(\log N)$ for fixed-entropy workloads); (2) each new expert monotonically improves hit rate; (3) a fleet of K cooperating units accumulates $K\times$ more observations per day, converging to full coverage proportionally faster than a single unit; (4) over-the-air (OTA) expert updates require $O(2^H \log \Delta N)$ bits per period, enabling $\approx 870\,\mathrm{KB/day}$ per robot for 1,000-unit fleets. We also prove *energy savings bounds*: cache hits consume $O(n + k\, d_{\text{model}})$ floating-point operations versus $O(Ln^2 + Lnd_{\text{model}})$ for a full forward pass, yielding up to $10^4\times$ energy reduction at high hit rates (Theorem 18).

We discuss deployment on personal computers, local robots, and vehicles—devices that can download only the experts they need on demand, learn from local workloads, upload observations to a shared repository, and receive certified domain intelligence as compact OTA updates. We situate the Safebox/Safebots.ai ecosystem [19] as one realization of this architecture, combining LAWS experts with hardware-attested policy execution and declarative orchestration.

Contents

1 Introduction **4**
- 1.1 The Problem: Repetition Without Reuse . . . 4
- 1.2 The Proposal: LAWS . . . 4
- 1.3 The Biological Parallel . . . 4
- 1.4 Overview of Contributions . . . 5

2 Background **6**
- 2.1 Probabilistic Language Tries . . . 6
- 2.2 Transformer Architecture . . . 7
- 2.3 Mixture of Experts . . . 7
- 2.4 Kahneman's Dual-Process Theory . . . 7
- 2.5 Chomsky's Innate Prior . . . 7
- 2.6 Symbolic AI: Cyc and Wolfram Alpha . . . 7

3 The Lipschitz Framework **8**
- 3.1 Component-wise Lipschitz Constants . . . 8

4 The LAWS Architecture **11**
- 4.1 Parametrized Experts . . . 11
- 4.2 Expert Function Classes . . . 11
- 4.3 LAWS Update Protocol . . . 12

5 Core Theorems **12**
- 5.1 Self-Certification . . . 12
- 5.2 Jacobian Correction . . . 13
- 5.3 Parameter Extraction at Branch Points . . . 13
- 5.4 Expert Library Dynamics . . . 14
- 5.5 LAWS as a Generalization . . . 16
- 5.6 Automatic Symbolic Vocabulary . . . 16

6 Parametrized Experts: Construction and Theory **17**
- 6.1 Pattern Recognition . . . 17
- 6.2 Small MLP Approximation . . . 18
- 6.3 Cross-Architecture Portability . . . 18

7 The Cinderella Effect **19**
- 7.1 The Problem . . . 19
- 7.2 The Cascade Bound . . . 19

8 Robotics and Fleet Learning **20**
- 8.1 The Robotics Challenge . . . 20
- 8.2 LAWS for Robotic Motor Programs . . . 20
- 8.3 State of the Art and LAWS Improvements . . . 21
- 8.4 Fleet Learning Theorems . . . 21
- 8.5 Over-the-Air (OTA) Expert Updates . . . 22

9 Comparison to Prior Work **23**
9.1 LAWS vs. KV Caching . . . 23
9.2 LAWS vs. Mixture of Experts . . . 23
9.3 LAWS vs. Cyc . . . 23
9.4 LAWS vs. Wolfram Alpha . . . 24

10 Conjectures and Open Problems **24**

11 Discussion **26**
11.1 LAWS for Diffusion Models . . . 26
11.2 Hardware Implementation . . . 26
11.3 LAWS as an AI Architecture Paradigm . . . 27
11.4 Limitations . . . 27

12 Discovering Laws: The Scientific Analogy **27**
12.1 Scientists Discover Laws; They Do Not Legislate Them . . . 27
12.2 Animals, Experts, and the System 1 Library . . . 28
12.3 Robotics and Vehicular Workloads as a Proving Ground . . . 28

13 Energy Savings and Edge Deployment **29**
13.1 The Energy Cost of Neural Inference . . . 29
13.2 Energy Cost of LAWS Cache Hits . . . 29
13.3 On-Demand Expert Download . . . 30

14 Safebots.ai and the Safebox Ecosystem **30**
14.1 Architecture Overview . . . 30
14.2 Tools, Policies, and Capabilities . . . 31
14.3 LAWS as the Inference Substrate for Distributed AI . . . 31

15 Conclusion **31**

1 Introduction

1.1 The Problem: Repetition Without Reuse

Modern neural networks are extraordinarily expensive to query. A forward pass through a 70B-parameter language model costs ∼140 TFLOP; a single diffusion image generation costs thousands of forward passes; a robot controller running at 100 Hz executes millions of forward passes per hour. Yet across all of these systems, the vast majority of queries are *not genuinely novel*. They are variations on patterns the model has processed before—the same algorithm in different variable names, the same manipulation task with a different object, the same legal clause with different parties.

Current systems exploit this redundancy only in the crudest ways. Key-value (KV) caches reuse computation for *exactly* repeated token prefixes. Mixture of Experts (MoE) routes tokens to specialized sub-networks, but the routing table is fixed at training time and never grows. Symbolic AI systems (Cyc, Wolfram Alpha) encode reusable patterns explicitly, but require human authors to write each one.

The fundamental gap: no existing system automatically discovers reusable computational patterns from deployment experience, certifies their validity formally, and grows the library continuously with use.

1.2 The Proposal: LAWS

We introduce *Learning from Actual Workloads Symbolically* (LAWS), an inference-time architecture that fills this gap.

LAWS sits above any base model as a transparent interception layer. It maintains a library $\mathcal{L}$ of *parametrized experts*, each encoding a cheap computation that approximates the base model on a certified region of input space. When a new query arrives:

1. A Probabilistic Language Trie (PLT) [15] lookup identifies all matching experts in $\mathcal{L}$ (those whose routing ball contains x).

2. If the query falls within e^*'s routing radius (distance $\leq \tau^*$, precomputed from model weights), LAWS returns e^*'s output—a cheap function evaluation, not a full forward pass.

3. Otherwise, the base model runs. Its output is recorded, potentially distilling a new expert into $\mathcal{L}$.

The routing radii are computed from the model's Lipschitz constant $\Lambda(W)$, computable from the trained weights without any inference. This is the *self-certification* property that distinguishes LAWS from all prior caching approaches.

1.3 The Biological Parallel

This architecture is not merely inspired by biology—it is a computational formalization of how biological intelligence actually works.

Kahneman's dual-process theory. Kahneman [11] describes two cognitive systems: System 1 (fast, automatic, pattern-based, requiring minimal cognitive effort) and System 2 (slow, deliberate, analytical, effortful). Expert performance in domains from chess to medicine to tennis consists largely of *System 1 operation*: the expert recognizes the situation as an instance of a known pattern

and executes the associated response without conscious deliberation. System 2 is invoked only when the situation is novel enough that no cached pattern applies.

LAWS formalizes this exactly. The expert library is System 1. The base model is System 2. The transition condition—query distance exceeds validity radius—is the abort-and-replan signal.

Chomsky's innate prior. Chomsky [7] argued that children acquire language too rapidly and too reliably from impoverished input for grammar to be learned from scratch. There must be an innate *Language Acquisition Device*—a prior structure in the brain that constrains the space of possible grammars.

In LAWS terms: the model's pre-trained weights W encode an innate prior. The PLT trie derived from W constitutes the model's innate expert library—the patterns it "knows" before seeing any deployment query. Real-world queries calibrate this prior, adding new experts and refining validity radii, but the innate structure is already rich at deployment time. The poverty-of-the-stimulus argument applies: the model can generalize to novel inputs immediately, without waiting for empirical frequency estimates, because the Lipschitz certificate derived from W defines valid generalization boundaries from the start.

The abort-and-replan signal. In air combat maneuvering, a missile engagement has a minimum operating range. Inside this range the seeker cannot track, the rocket motor may not have armed, and the geometry is too dynamic for guidance. The pilot's abort signal is immediate and automatic: *too close for missiles, switching to guns.* No deliberation. The abort condition was precomputed during mission planning; the in-flight check is $O(1)$.

LAWS's transition from System 1 to System 2 is the same operation. The validity radius τ^* is precomputed from W at model-load time. At query time, a single comparison—does $d_{\mathcal{T}}(x, e^*) \leq \tau^*$?—routes the query. The decision requires no additional inference.

1.4 Overview of Contributions

1. **Transformer Lipschitz Bound** (Theorem 1): $\Lambda(W)$ is computable from model weights; activations are Lipschitz in token embeddings with this constant.

2. **Self-Certification Theorem** (Theorem 3): validity of every LAWS expert is certified by $\Lambda(W)$ without inference.

3. **Jacobian Correction** (Theorem 4): parametrized experts achieve $O(r^2)$ error (vs. $O(r)$ for exact-match caches) where $r = \|E(x_{\bar{d}+1}) - E(n^*_{\bar{d}+1})\|$, via precomputed Jacobians at the divergence embedding.

4. **Expert Library Growth Rate** (Theorem 7): under stationary $P_{\mathcal{M}}$, new expert creation rate is $O(2^{H(P_{\mathcal{M}})} \log N)$ after N queries; $O(\log N)$ for fixed-entropy workloads.

5. **Monotone Hit Rate** (Theorem 6): each new expert weakly increases the expected hit rate for all future queries.

6. **Abort-and-Replan Threshold** (Theorem 9): a unique τ^* minimizes expected inference cost; computable from $\Lambda(W)$ and cost parameters.

7. **LAWS Generalizes MoE and KV Caching** (Theorem 10): both are recovered as degenerate cases; LAWS is strictly more expressive.

8. **Fleet Learning Lower Bound** (Theorem 15): K cooperating LAWS units achieve hit rate improvement $\Omega(K)$ over single-unit deployment.

9. **Automatic Symbolic Vocabulary** (Theorem 11): the PLT trie of a trained model defines a graded, compositional, self-certified symbolic vocabulary without human authorship.

10. **Cinderella Cascade Bound** (Theorem 14): amplification of dropped attention weights through layers is bounded by the surprisal of the dropped token; rare high-surprisal events constitute the Shannon overflow set.

11. **Robotics Convergence Rate** (Corollary of Theorem 15): for a fleet of K robots each performing M tasks per day, LAWS expert library coverage of the task distribution converges at rate $\Omega(KM)$.

12. **OTA Download Bound** (Theorem 16): the incremental expert library update for period $[t, t+\Delta t]$ has description length $O(2^{H(P_{\mathcal{M}})} \cdot \log(\Delta N) \cdot B_{\text{expert}})$ bits, characterizing the bandwidth required for over-the-air updates.

13. **Energy Savings Bound** (Theorem 18): at asymptotic hit rate H_∞, LAWS reduces energy per query to $H_\infty \cdot O(n+k\,d_{\text{model}})/O(Ln^2+Lnd_{\text{model}}) + (1 - H_\infty)$ of baseline; $\sim 10\times$ reduction at 90% hit rate for typical LLM parameters.

14. **Conjecture: Symbolic Pattern Emergence** (Conjecture 3): for a sufficiently capable base model, all high-probability trie nodes have experts in one of a finite set of primitive function classes.

Organization. Section 2 reviews PLTs, MoE, KV caching, and cognitive science background. Section 3 establishes the Lipschitz framework. Section 4 defines LAWS formally. Section 5 contains the core theorems and proofs. Section 6 develops the parametrized expert framework. Section 7 treats the Cinderella cascade effect. Section 8 develops the robotics application with fleet-learning theorems. Section 9 compares to Cyc, Wolfram Alpha, MoE, and KV caching. Section 11 discusses diffusion models, hardware, and open problems.

2 Background

2.1 Probabilistic Language Tries

We briefly recall the framework of [15]. Let V be a finite vocabulary and $\mathcal{M}$ a generative model over V^*.

Definition 1 (PLT and Trie Metric [15])**.** *The* probabilistic language trie $\mathcal{T}(\mathcal{M})$ *is the directed rooted tree whose nodes are prefixes $x \in V^*$ and whose outgoing edges from x are labeled by tokens $t \in V$ with weight $P_{\mathcal{M}}(t \mid x)$. For two sequences $s, s' \in V^*$, their* longest common prefix *is $s \wedge s'$, and the* trie metric *is:*

$$d_{\mathcal{T}}(s, s') = -\log_2 P_{\mathcal{M}}(s \wedge s').$$

The companion paper on sequential KV compression [17] established that, for a transformer $\mathcal{M}$, the conditional entropy of KV vectors satisfies:

$$H(\text{KV}_{t+1} \mid \text{KV}_{\le t}) \;\le\; H(t_{t+1} \mid t_1, \ldots, t_t) \;\le\; \log_2 \text{PP}(\mathcal{M}),$$

bounding KV cache entropy by per-token surprisal. LAWS builds on this result: the same surprisal that bounds KV entropy also bounds the error in a parametrized expert approximation.

2.2 Transformer Architecture

A standard pre-norm transformer with L layers processes a token sequence $\mathbf{t} = (t_1, \ldots, t_n)$ via a residual stream:

$$\mathbf{x}_i^{(\ell+1)} = \mathbf{x}_i^{(\ell)} + \mathrm{Attn}^{(\ell)}\big(\mathrm{LN}(\mathbf{x}^{(\ell)})\big)_i + \mathrm{MLP}^{(\ell)}\big(\mathrm{LN}(\mathbf{x}^{(\ell+1)\ (\mathrm{partial})})\big)_i.$$

The KV vectors at layer ℓ, position i are $\mathbf{k}_i^{(\ell)} = W_K^{(\ell)}\mathrm{LN}(\mathbf{x}_i^{(\ell)})$ and $\mathbf{v}_i^{(\ell)} = W_V^{(\ell)}\mathrm{LN}(\mathbf{x}_i^{(\ell)})$. The full forward map is $F_W : V^n \to \mathbb{R}^{|V|}$.

2.3 Mixture of Experts

Definition 2 (MoE [20])**.** *A* Mixture of Experts *layer maintains K expert networks $\{e_1, \ldots, e_K\}$ and a router $R : \mathbb{R}^d \to \Delta^K$. For input x, the output is $\sum_{k=1}^{K} R(x)_k \cdot e_k(x)$ (or $\sum_{k \in \text{top-}k'} R(x)_k \cdot e_k(x)$ in sparse variants).*

Key properties of standard MoE: (1) K is fixed at training time; (2) experts are trained jointly with the router; (3) no formal guarantee that any expert is correct on any given input.

2.4 Kahneman's Dual-Process Theory

Kahneman [11] identifies two cognitive systems. *System 1* is fast, automatic, and based on pattern recognition; it operates below conscious awareness and requires minimal effort. *System 2* is slow, deliberate, and analytical; it requires working memory and conscious engagement. Expert performance—in chess, medicine, tennis, piloting—consists overwhelmingly of System 1 operation: the expert pattern-matches the current situation to a stored schema and executes the associated response automatically. System 2 is invoked when the situation is genuinely novel: when no stored pattern applies, or when the System 1 response triggers a conflict signal.

This migration from System 2 to System 1 with experience is well-documented. A novice chess player consciously evaluates candidate moves; a grandmaster perceives the "right" move immediately. A medical student follows diagnostic algorithms; an experienced clinician recognizes the disease pattern from a symptom constellation. The expertise lies in the richness and accuracy of the System 1 library, not in faster System 2 processing.

2.5 Chomsky's Innate Prior

Chomsky's *poverty of the stimulus* argument [7] observes that children acquire complex grammar from limited, often ungrammatical input. The input alone is insufficient to determine a unique grammar; children must bring prior structure to the task. This prior structure—the *Language Acquisition Device* (LAD)—constrains the space of possible grammars the child considers.

In our framework: a trained model's weights W are its LAD. The PLT trie derived from W is the innate expert library—the patterns the model "pre-knows" before any deployment query. As in Chomsky's theory, this prior is not learned from deployment observations; it is encoded in W during pre-training on a large corpus. The self-certification theorem (Theorem 3) formalizes the sense in which W certifies the validity of this innate knowledge.

2.6 Symbolic AI: Cyc and Wolfram Alpha

Cyc [14]. The Cyc project, begun in 1984, aimed to encode human common-sense knowledge as explicit logical axioms. After approximately 47,000 person-years of manual knowledge entry, the

system contains roughly 25 million rules and can perform logical inference over them. The fundamental limitation: every piece of knowledge must be hand-authored by a human who understands it. Cyc's knowledge base is static between manual updates. It provides no graceful degradation for inputs outside the authored rules, and no formal guarantee that any rule is correct.

Wolfram Alpha [22]. Wolfram Alpha takes a different approach: curated computational knowledge (mathematical functions, physical constants, geographic data) combined with a symbolic computation engine. It is extremely powerful within its curated domain, but the domain boundary is sharp. Outside curated knowledge, the system fails. Like Cyc, knowledge is manually structured by Wolfram and his team.

LAWS vs. symbolic AI. LAWS differs from both in three fundamental respects: (1) its symbolic vocabulary is discovered automatically from the trained model's distribution, not authored by humans; (2) it provides formal validity certificates for every expert, derived from W; and (3) its vocabulary grows continuously with deployment without human intervention. The full comparison is in Section 9.

3 The Lipschitz Framework

3.1 Component-wise Lipschitz Constants

We establish Lipschitz constants for each component of a transformer layer.

Lemma 1 (Lipschitz Constants of Transformer Components)**.** *Let $\gamma^{(\ell)}, \varepsilon_{\text{LN}} > 0$ be the LayerNorm scale and stability constant at layer ℓ, and let $\|\cdot\|_{\text{op}}$ denote operator norm. The following hold:*

(a) ***LayerNorm:*** $\|\text{LN}^{(\ell)}(x) - \text{LN}^{(\ell)}(y)\| \le (\gamma^{(\ell)}/\varepsilon_{\text{LN}})\|x - y\|$ *for all x, y with* $\min(\text{Var}(x), \text{Var}(y)) \ge \varepsilon_{\text{LN}}$.

(b) ***Linear projection:*** $\|Wx - Wy\| \le \|W\|_{\text{op}}\|x - y\|$.

(c) ***Softmax attention:*** *the attention output map $x \mapsto \sum_j \text{softmax}(Qx \cdot K/\sqrt{d_{\text{head}}})_j V_j$ is Lipschitz with constant $L_{\text{attn}}^{(\ell)} \le \|W_V^{(\ell)}\|_{\text{op}} + \frac{1}{2\sqrt{d_{\text{head}}}}\|W_Q^{(\ell)}\|_{\text{op}}\|W_K^{(\ell)}\|_{\text{op}} \cdot \max_j \|\mathbf{v}_j\|$.*

(d) ***GeLU/ReLU:*** *L_{act}-Lipschitz, where $L_{\text{act}} = 1$ for ReLU and $L_{\text{act}} \approx 1.083$ for GeLU; in both cases L_{act} is absorbed into $\kappa^{(\ell)}$.*

Proof. (a) LayerNorm computes $\text{LN}(x) = \gamma \cdot (x - \mu(x))/\sigma(x)$ where μ is the mean and σ is the standard deviation. The map $x \mapsto (x - \mu)/\sigma$ is differentiable with Jacobian bounded in operator norm by $1/\sigma_{\min} \le 1/\varepsilon_{\text{LN}}$ when $\sigma(x) \ge \varepsilon_{\text{LN}}$. Multiplying by γ gives (a).

(b) Immediate from the definition of operator norm.

(c) The attention output at position i is $o_i = \sum_j a_{ij}\mathbf{v}_j$ where $a_{ij} = \text{softmax}(\mathbf{q}_i \cdot \mathbf{k}_j/\sqrt{d_{\text{head}}})_j$. The total Lipschitz constant has two components: (i) changes in o_i due to changes in the value vectors $\mathbf{v}_j = W_V^{(\ell)}\mathbf{x}_j$, bounded by $\|W_V^{(\ell)}\|_{\text{op}}$ (since $\sum_j a_{ij} = 1$ and $a_{ij} \ge 0$); and (ii) changes in o_i due to changes in the attention weights a_{ij}, which depend on the query $\mathbf{q}_i = W_Q^{(\ell)}\mathbf{x}_i$. The softmax function $\sigma : \mathbb{R}^n \to \Delta^{n-1}$ satisfies $\|\sigma(u) - \sigma(v)\|_1 \le \|u - v\|_1$ (Lipschitz constant 1 in ℓ^1) and $\|\sigma(u) - \sigma(v)\|_2 \le \frac{1}{2}\|u - v\|_2$. Differentiating o_i with respect to the query $\mathbf{q}_i$ gives $\partial o_i/\partial \mathbf{q}_i = \frac{1}{\sqrt{d_{\text{head}}}}\sum_j (\partial a_{ij}/\partial \mathbf{q}_i)\mathbf{v}_j^T$. The Jacobian of the attention weights satisfies $\|\partial a_i/\partial \mathbf{q}_i\|_{\text{op}} \le \|W_K^{(\ell)}\|_{\text{op}}/(2\sqrt{d_{\text{head}}})$, giving the stated

bound by summing components (i) and (ii). *Note:* when position $\bar{d}+1$'s query changes, it attends to all n key vectors; a precise bound acquires a factor of $\sqrt{n}$ from the Cauchy–Schwarz step $\|\delta\text{logit}\|_2 \leq \sqrt{n} \cdot \max_j |\delta\text{logit}_j|$. The stated L_{attn} is therefore context-length-dependent; $\Lambda(W)$ should be understood as $\Lambda(W, n)$ in long-context settings.

(d) GeLU satisfies $\text{GeLU}(x) = x\Phi(x)$, where Φ is the standard Gaussian CDF. Its derivative is $\text{GeLU}'(x) = \Phi(x) + x\phi(x)$, where ϕ is the Gaussian PDF. The maximum of $|\text{GeLU}'|$ is achieved at $x \approx 1.1$ where $\text{GeLU}'(x) \approx 1.083$. *Technically, GeLU is not 1-Lipschitz; it is L_{GeLU}-Lipschitz with $L_{\text{GeLU}} \approx 1.1$.* We incorporate this constant into $\kappa^{(\ell)}$ via the MLP Lipschitz factor. SwiGLU and ReLU variants are handled analogously; ReLU is exactly 1-Lipschitz. For practical purposes, this constant is absorbed into $\|W^{(\ell)}_{\text{MLP}}\|_{\text{op}}$ in Theorem 1, which we define as including activation Lipschitz constants. □

Theorem 1 (Transformer Lipschitz Bound)**.** *Let $F_W : V^n \to \mathbb{R}^{|V|}$ be a transformer with pre-norm architecture. Define the* layer coupling constant*:*

$$\kappa^{(\ell)} = \left(1 + L^{(\ell)}_{\text{attn}} \cdot \frac{\gamma^{(\ell)}}{\varepsilon_{\text{LN}}}\right) \cdot \left(1 + \|W^{(\ell)}_{\text{MLP}}\|_{\text{op}} \cdot \frac{\gamma^{(\ell)}}{\varepsilon_{\text{LN}}}\right),$$

and the end-to-end Lipschitz constant*:*

$$\Lambda(W) = \prod_{\ell=1}^{L} \kappa^{(\ell)}.$$

Then for any two token sequences s, s' that diverge first at position $\bar{d}+1$:

$$\|F_W(s) - F_W(s')\| \;\leq\; \Lambda(W) \cdot \|E(s_{\bar{d}+1}) - E(s'_{\bar{d}+1})\|,$$

where $E : V \to \mathbb{R}^{d_{\text{model}}}$ is the token embedding. $\Lambda(W)$ is computable from the trained weights in $O(L \cdot d^2_{\text{model}})$ operations.

Proof. For positions $i \leq \bar{d}$, the sequences are identical, so all activations $\mathbf{x}^{(\ell)}_i(s) = \mathbf{x}^{(\ell)}_i(s')$ exactly (by determinism of the forward pass—Lemma 1 of [17]).

At position $\bar{d}+1$, the difference in activations is introduced at the embedding layer: $\|\mathbf{x}^{(0)}_{\bar{d}+1}(s) - \mathbf{x}^{(0)}_{\bar{d}+1}(s')\| = \|E(s_{\bar{d}+1}) - E(s'_{\bar{d}+1})\|$.

At positions $i > \bar{d}+1$: through causal attention, the perturbation at position $\bar{d}+1$ propagates to all later positions. At each layer ℓ, every position's hidden state change is bounded by $\kappa^{(\ell)}$ times the maximum change from the previous layer, since the layer map is $\kappa^{(\ell)}$-Lipschitz over the full n-position activation tensor. Therefore the bound $\kappa^{(\ell)} \cdot \|\delta^{(\ell-1)}\|_{\max}$ applies uniformly across all positions at each layer.

Each transformer layer is a composition of the components in Lemma 1. By the chain rule for Lipschitz constants, the Lipschitz constant of the composition is at most the product of the individual constants, giving $\kappa^{(\ell)}$ per layer. Chaining across all L layers gives $\Lambda(W)$.

Computability: $\Lambda(W)$ requires computing operator norms of weight matrices, each taking $O(d^2_{\text{model}})$ operations via the power method. Summing over L layers gives $O(L \cdot d^2_{\text{model}})$. □

Remark 1 (On the Tightness of $\Lambda(W)$)**.** *$\Lambda(W)$ can be large for deep networks (exponential in L in the worst case), since it is a product of per-layer operator norms. Three caveats apply: (1) The LayerNorm bound requires $\sigma(x) \geq \varepsilon_{\text{LN}}$ (input variance bounded away from zero), which holds throughout*

trained transformers due to the ε_{LN} stabilizer in the denominator, but may fail for adversarially constructed inputs outside the training distribution. We restrict LAWS validity claims to inputs drawn from $P_{\mathcal{M}}$, for which this holds with high probability. (2) In practice, the effective *Lipschitz constant on in-distribution inputs is far smaller than $\Lambda(W)$; empirical calibration is recommended for setting τ^* in production systems. (3) LAWS uses $\Lambda(W)$ in the self-certification bound $\varepsilon_{\text{fit}} + 2\,\Lambda(W) \cdot C_E$; for this to be a useful guarantee requires $\delta > \varepsilon_{\text{fit}} + 2\,\Lambda(W) \cdot C_E$, i.e., $\Lambda(W) < (\delta - \varepsilon_{\text{fit}})/(2C_E)$. (4) As noted in Lemma 1(c), the attention Lipschitz constant acquires a $\sqrt{n}$ factor from the query-to-all-keys term; for long contexts, $\Lambda(W)$ should be treated as $\Lambda(W, n)$ scaling with sequence length.*

Corollary 1 (Activation Similarity Within Shared Prefix)**.** *For any two inputs sharing a prefix of length $\bar{d}$ and any layer ℓ, position $i \le \bar{d}$:*

$$\|\mathbf{x}_i^{(\ell)}(s) - \mathbf{x}_i^{(\ell)}(s')\| = 0.$$

For position $i = \bar{d} + 1$ and layer ℓ:

$$\|\mathbf{x}_{\bar{d}+1}^{(\ell)}(s) - \mathbf{x}_{\bar{d}+1}^{(\ell)}(s')\| \le \prod_{m=1}^{\ell} \kappa^{(m)} \cdot \|E(s_{\bar{d}+1}) - E(s'_{\bar{d}+1})\|.$$

Proof. The first statement follows from KV determinism [17]. The second follows from the proof of Theorem 1 truncated at layer ℓ. □

Theorem 2 (Lower Bound on $\Lambda(W)$ for Structured Tasks)**.** *Let $\mathcal{T}$ be a task class requiring the base model to produce outputs whose pairwise ℓ^2 separation is at least $\Delta > 0$ for inputs whose divergence embeddings satisfy $\|E(s_{\bar{d}+1}) - E(s'_{\bar{d}+1})\| \le r$. Then:*

$$\Lambda(W) \;\ge\; \frac{\Delta}{r}.$$

In particular, for arithmetic tasks where the model must distinguish outputs differing by at least $\Delta = 1/M$ (decimal precision M), the bound gives $\Lambda(W) \ge \Delta / r_{\min}$, where $r_{\min} = \min_{t \ne t'} \|E(t) - E(t')\|$ is the minimum pairwise embedding separation (a property of the trained embedding matrix, measurable directly from W). For tasks where semantically adjacent tokens (e.g., consecutive digits "1" and "2") have small embedding separation $r_{\min} \ll C_E$, the bound forces $\Lambda(W) \ge 1/(M \cdot r_{\min})$, which grows with precision M. High-precision arithmetic thus provably requires larger $\Lambda(W)$, limiting the LAWS routing radius τ^* for such tasks.

Proof. By the definition of the Lipschitz constant: $\|F_W(s) - F_W(s')\| \le \Lambda(W) \cdot \|E(s_{\bar{d}+1}) - E(s'_{\bar{d}+1})\|$ for any pair diverging at $\bar{d} + 1$. If there exist inputs s, s' with $\|E(s_{\bar{d}+1}) - E(s'_{\bar{d}+1})\| \le r$ and $\|F_W(s) - F_W(s')\| \ge \Delta$, then $\Lambda(W) \ge \Delta / r$. Such inputs exist whenever the model must distinguish outputs separated by Δ from embeddings separated by only r, which is exactly the case for high-precision arithmetic or lookup tasks where adjacent vocabulary tokens map to semantically distant outputs. □ □

Corollary 2 (LAWS Acceleration Limits for High-Precision Tasks)**.** *For any task class requiring output precision Δ from minimal embedding perturbation r, the LAWS routing radius satisfies:*

$$\tau^* \;\le\; \frac{\delta - \varepsilon_{\text{fit}} - 2(\Delta/r) \cdot C_E}{(\Delta/r) \cdot C_E}.$$

If $\Delta > r(\delta - \varepsilon_{\text{fit}})/(2C_E)$ (high required precision relative to the quality threshold), then $\tau^* < 0$ and no expert can be certified: LAWS must always invoke the base model for such tasks. This characterizes the fundamental boundary *of LAWS acceleration—it cannot certify experts for tasks that require the model to be highly sensitive to small input perturbations.*

4 The LAWS Architecture

4.1 Parametrized Experts

Definition 3 (Parametrized Expert)**.** *A* parametrized expert *is a tuple* $e = (n^*, f, \phi, \tau^*, \varepsilon_{\text{fit}})$ *where:*

- $n^* \in \mathcal{T}(\mathcal{M})$ *is the* signpost*: a full-length input in* V^n*, represented as a PLT trie node (by its token-prefix structure);* $F_W(n^*)$ *is the base model output on this input;*
- $\phi : V^* \to \mathbb{R}^k$ *is the* parameter extractor*, mapping inputs to a* k*-dimensional parameter vector;*
- $f : \mathbb{R}^k \to \mathbb{R}^{|V|}$ *is the* expert function*, a cheap computation approximating* F_W *on the expert's domain;*
- $\tau^* > 0$ *is the* validity radius *in PLT metric space, derived from* $\Lambda(W)$*;*
- $\varepsilon_{\text{fit}} \geq 0$ *is the* fitting error*, certifying* $\|F_W(n^*) - f(\phi(n^*))\| \leq \varepsilon_{\text{fit}}$*.*

The validity domain *of* e *is the ball* $\mathcal{B}(n^*, \tau^*) = \{x \in V^* : d_{\mathcal{T}}(x, n^*) \leq \tau^*\}$*.*

Definition 4 (Expert Library and LAWS Inference)**.** *A* LAWS system $(F_W, \mathcal{L}, \delta)$ *consists of a base model* F_W*, a library* $\mathcal{L}$ *of parametrized experts, and a quality threshold* $\delta > 0$ *satisfying:*

$$\delta \;>\; \varepsilon_{\text{fit}} + \Lambda(W) \cdot C_E \cdot (2 + H(P_{\mathcal{M}})),$$

where $H(P_{\mathcal{M}})$ *is the entropy of the input distribution. This ensures (i) the self-certification bound* $\varepsilon_{\text{fit}} + 2\Lambda C_E \leq \delta$ *(Theorem 3), and (ii) the routing radius* $\tau^* \geq H$*, so that every heavy trie node* n^* *(with* $P_{\mathcal{M}}(n^*) \geq 2^{-H}$*) lies within its own routing ball under any-match routing. The* routing radius *for an expert is:*

$$\tau^*(n^*, \delta) = \frac{\delta - \varepsilon_{\text{fit}} - 2\,\Lambda(W) \cdot C_E}{\Lambda(W) \cdot C_E}$$

where $C_E = \max_{t,t'} \|E(t) - E(t')\|$*. Any expert is correct on all inputs; the routing radius defines which queries this expert handles.*

For input x*,* LAWS inference *proceeds as:*

$$\text{LAWS}(x) = \begin{cases} f_{e^*}(\phi_{e^*}(x)) & \text{if } \exists e \in \mathcal{L} : d_{\mathcal{T}}(x, n_e^*) \leq \tau^*(e) \\ F_W(x) & \text{otherwise (cache miss)} \end{cases}$$

where $e^* = \arg\min_{e : d_{\mathcal{T}}(x, n_e^*) \leq \tau^*(e)} d_{\mathcal{T}}(x, n_e^*)$ *(the matching expert with smallest trie distance; a cache hit occurs whenever* any *expert's routing ball contains* x*, not only the globally nearest expert). This "any-match" rule ensures that adding experts can only increase coverage, never decrease it—a property necessary for Theorem 6.*

4.2 Expert Function Classes

Expert functions f may be chosen from an increasing hierarchy of complexity:

Level 1. Constant. $f(\phi) = c$ where $c = F_W(n^*)$. Standard exact-match cache. Error $\leq \Lambda(W) \cdot C_E$ for all x (Term 1 of Theorem 3 only; Terms 2 and 3 are zero for a constant expert).

Level 2. Linear (Jacobian correction). $f(\phi) = F_W(n^*) + J_W(n^*) \cdot \phi$ where J_W is the Jacobian of F_W at n^*. Error $O(r^2)$ where $r = \|E(x_{\bar{d}+1}) - E(n^*_{\bar{d}+1})\|$ (Theorem 4).

Algorithm 1 LAWS Update

Require: Input x, output $y = F_W(x)$, library $\mathcal{L}$, threshold $N_{\min}$

$n \leftarrow \mathcal{T}(\mathcal{M}).\text{insert}(x, y)$ ▷ Update trie with observation
if $n.\text{count} \geq N_{\min}$ **then**
  $(f, \phi, \varepsilon_{\text{fit}}) \leftarrow \text{PatternRecognize}(n.\text{samples})$
  $\tau^* \leftarrow (\delta - \varepsilon_{\text{fit}} - 2\,\Lambda(W) \cdot C_E)/(\Lambda(W) \cdot C_E)$
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(n, f, \phi, \tau^*, \varepsilon_{\text{fit}})\}$
end if

Level 3. Primitive function. f is an identified algorithmic pattern: arithmetic, sorting, lookup table, string template, JSON/SQL instantiation. Error zero (exact).

Level 4. Small MLP. f is a small neural network (2–3 layers, width ≤ 128) fit to the signpost's subtree. Error ε_{fit} by construction.

4.3 LAWS Update Protocol

5 Core Theorems

5.1 Self-Certification

Theorem 3 (Self-Certification)**.** *For any LAWS expert $e = (n^*, f, \phi, \tau^*, \varepsilon_{\text{fit}})$ satisfying Definition 3, with ϕ extracting the parameter vector at the divergence embedding (so $\|\phi(x) - \phi(n^*)\| \leq C_E$ for any x), and with $\text{Lip}(f) \leq \Lambda(W)$, the approximation error is bounded* uniformly *over all inputs x:*

$$\|F_W(x) - f(\phi(x))\| \;\leq\; \varepsilon_{\text{fit}} + 2\,\Lambda(W) \cdot C_E.$$

In particular, if $\delta > \varepsilon_{\text{fit}} + 2\,\Lambda(W) \cdot C_E$, then $\|F_W(x) - f(\phi(x))\| \leq \delta$ for every *input x—no restriction on $d_{\mathcal{T}}(x, n^*)$ is required. The validity radius τ^* in Definition 4 bounds the validity domain for routing purposes (to select the best expert), not for correctness.*

Proof. By the triangle inequality:

$$\|F_W(x) - f(\phi(x))\| \leq \underbrace{\|F_W(x) - F_W(n^*)\|}_{\text{Term 1}} + \underbrace{\|F_W(n^*) - f(\phi(n^*))\|}_{\text{Term 2}} + \underbrace{\|f(\phi(n^*)) - f(\phi(x))\|}_{\text{Term 3}}.$$

Term 2 is $\leq \varepsilon_{\text{fit}}$ by the fitting condition.

Term 3. By assumption $\|\phi(n^*) - \phi(x)\| \leq C_E$ and $\text{Lip}(f) \leq \Lambda(W)$: for Level-1 experts $\text{Lip}(f) = 0$; for Level-2 Jacobian experts, $\text{Lip}(f) = \|J_W(n^*)\|_{\text{op}} \leq \Lambda(W)$ by Theorem 1; for small MLPs verified at fitting time. Therefore Term 3 $\leq \Lambda(W) \cdot C_E$.

Term 1. Let $\bar{d}$ be the length of the longest common token prefix of x and n^*. By Theorem 1:

$$\text{Term 1} \leq \Lambda(W) \cdot \|E(x_{\bar{d}+1}) - E(n^*_{\bar{d}+1})\| \leq \Lambda(W) \cdot C_E.$$

Summing:

$$\|F_W(x) - f(\phi(x))\| \leq \varepsilon_{\text{fit}} + \Lambda(W) \cdot C_E + \Lambda(W) \cdot C_E = \varepsilon_{\text{fit}} + 2\,\Lambda(W) \cdot C_E \leq \delta. \qquad \square$$

$\square$

5.2 Jacobian Correction

Theorem 4 (Jacobian Correction Achieves Second-Order Error)**.** *Let e be a Level-2 expert with $f(\phi) = F_W(n^*) + J_W(n^*) \cdot \phi$ and $\phi(x) = E(x_{\bar{d}+1}) - E(n^*_{\bar{d}+1})$ where $\bar{d}$ is the length of the longest common prefix of x and n^*. Define $r = \|E(x_{\bar{d}+1}) - E(n^*_{\bar{d}+1})\|$ as the embedding-space distance at the divergence point. Then:*

$$\|F_W(x) - f(\phi(x))\| \;\le\; \frac{1}{2}\|H_W(n^*)\|_{\text{op}} \cdot r^2 + O(r^3)$$

where $H_W(n^*)$ is the Hessian of F_W at n^* with respect to the divergence embedding. In particular the error is $O(r^2)$ versus $O(r)$ for a Level-1 (constant) expert, giving a strict improvement for $r < 1$.

Proof. Since x and n^* share the first $\bar{d}$ tokens, all activations at positions $\le \bar{d}$ are identical. The only perturbation is $\mathbf{u} = E(x_{\bar{d}+1}) - E(n^*_{\bar{d}+1})$ at the divergence position, with $\|\mathbf{u}\| = r$. By Taylor's theorem applied to F_W as a function of the divergence embedding (the forward pass is smooth in the embedding inputs):

$$F_W(x) = F_W(n^*) + J_W(n^*)\,\mathbf{u} + \frac{1}{2}\,\mathbf{u}^T H_W(\xi)\,\mathbf{u}$$

for some ξ on the segment between $E(n^*_{\bar{d}+1})$ and $E(x_{\bar{d}+1})$. The Level-2 expert returns $F_W(n^*) + J_W(n^*)\phi(x) = F_W(n^*) + J_W(n^*)\,\mathbf{u}$, so the error equals the remainder, bounded by $\frac{1}{2}\|H_W(\xi)\|_{\text{op}}\, r^2$. By continuity of the Hessian on the compact validity ball, $\|H_W(\xi)\|_{\text{op}} \le \|H_W(n^*)\|_{\text{op}} + O(r)$, giving the stated bound. □

Corollary 3 (Quadratic Improvement Over Standard Cache)**.** *For $r = \|E(x_{\bar{d}+1}) - E(n^*_{\bar{d}+1})\| < 1$, the Level-2 expert improves on the Level-1 expert by a factor of r in error. For $r = 0.1$, the Jacobian correction reduces error by a factor of 10; for $r = 0.01$, by a factor of 100.*

5.3 Parameter Extraction at Branch Points

Theorem 5 (Parameters Lie at High-Entropy Positions)**.** *Let n^* be a trie node at depth d_* (token prefix length) and let positions $\mathcal{P}_{\text{hi}} = \{i > d_* : H(t_i \mid t_{<i}) > H_{\text{thresh}}\}$ be the high-entropy positions after the shared prefix. Define the parameter extractor $\phi_{\text{hi}}(x) = (E(x_i))_{i \in \mathcal{P}_{\text{hi}}}$.*

Then the expected *contribution of a low-entropy position $j \notin \mathcal{P}_{\text{hi}}$ to the expert error satisfies:*

$$\mathbb{E}\big[\Lambda(W) \cdot \|E(t_j) - \mathbb{E}[E(t_j) \mid t_{<j}]\|\big] \le \Lambda(W) \cdot C_E \cdot \sqrt{H(t_j \mid t_{<j}) \cdot \ln 2} < \Lambda(W) \cdot C_E \cdot \sqrt{H_{\text{thresh}} \cdot \ln 2},$$

which is below the resolution ε_{fit} when $H_{\text{thresh}} < (\varepsilon_{\text{fit}}/(\Lambda(W) \cdot C_E))^2 / \ln 2$. Therefore, ϕ_{hi} is sufficient on average: in expectation over queries from $P_{\mathcal{M}}$, low-entropy positions can be ignored by the parameter extractor without increasing expected error above δ.

Proof. By the surprisal-bounded embedding variance result of [17] (the variance–entropy coupling result, Corollary on coherent text):

$$\mathbb{E}\big[\|E(t_j) - \mathbb{E}[E(t_j) \mid t_{<j}]\|^2\big]^{1/2} \le C_E \sqrt{H(t_j \mid t_{<j}) \cdot \ln 2}.$$

For positions with $H(t_j \mid t_{<j}) < H_{\text{thresh}}$, the token is nearly determined by the context, so $E(t_j) \approx \mathbb{E}[E(t_j) \mid t_{<j}]$ with deviation bounded by $C_E\sqrt{H_{\text{thresh}} \cdot \ln 2}$. By Theorem 1, this deviation propagates to output error $\le \Lambda(W) \cdot C_E \sqrt{H_{\text{thresh}} \cdot \ln 2}$. Setting this below ε_{fit} gives the threshold condition. □

5.4 Expert Library Dynamics

Theorem 6 (Monotone Hit Rate)**.** *Let H_n be the expected LAWS cache hit rate after n deployment queries drawn i.i.d. from $P_{\mathcal{M}}$. Under the LAWS update protocol (Algorithm 1), H_n is non-decreasing in n:*

$$H_1 \leq H_2 \leq \cdots \leq H_n \leq \cdots \leq 1.$$

Proof. Under the any-match inference rule, a query x is a cache hit whenever *any* expert's routing ball contains x. Adding a new expert (n^*, f, ϕ, τ^*) adds $\mathcal{B}(n^*, \tau^*)$ to the union of all routing balls. Since this union can only grow (never shrink) when experts are added, $H_{n+1} \geq H_n$. The inequality is strict whenever $\mathcal{B}(n^*, \tau^*)$ contains any x not already covered by prior routing balls—guaranteed since the triggering query x was a miss (outside all prior routing balls) and $P_{\mathcal{M}}(\{x\}) > 0$. □

Theorem 7 (Expert Library Growth Rate)**.** *Under a stationary input distribution $P_{\mathcal{M}}$ with Shannon entropy $H = H(P_{\mathcal{M}})$, and with the LAWS update threshold $N_{\min}$ (minimum observations before creating an expert), the expected number of new LAWS experts created after N queries satisfies:*

$$\mathbb{E}[\textit{new experts after } N \textit{ queries}] = O\big(2^H\big),$$

provided $N \leq N_{\min} \cdot 2^H$ (so that only heavy *trie nodes accumulate enough observations to trigger expert creation). The tight bound is $O(2^H)$ throughout this regime; the weaker form $O(2^H \log N)$ is used in later results to simplify expressions involving N explicitly (since $2^H \leq 2^H \log N$ for $N \geq 2$).*

Proof. New experts are created only on cache misses, and only when a trie node accumulates $N_{\min}$ observations (Algorithm 1).

Partition trie nodes into *heavy* ($P_{\mathcal{M}}(n) \geq \varepsilon$) and *light* ($P_{\mathcal{M}}(n) < \varepsilon$). Set $\varepsilon = 2^{-H}$.

Heavy nodes. By the AEP, $|\mathcal{H}_\varepsilon| \leq 2^H$. The expected number of distinct heavy nodes visited in N draws is at most $\min(N, 2^H) \leq 2^H$ (occupancy bound, as below). Each creates at most one expert, giving $\mathbb{E}[\text{heavy experts}] \leq 2^H$.

Light nodes. A light node n has $P_{\mathcal{M}}(n) < 2^{-H}$, so its expected visits in N queries is $N \cdot P_{\mathcal{M}}(n) < N \cdot 2^{-H}$. To trigger expert creation it needs $N_{\min}$ visits. If $N \leq N_{\min} \cdot 2^H$, then $N \cdot P_{\mathcal{M}}(n) < N_{\min}$ in expectation, so no light node accumulates enough visits to create an expert. Therefore $\mathbb{E}[\text{light experts}] = 0$ in this regime. ✓

Occupancy bound. For any M elements with probabilities $p_1, \ldots, p_M$:

$$\mathbb{E}[\text{distinct elements in } N \text{ draws}] = \sum_{i=1}^{M} \big(1 - (1 - p_i)^N\big) \leq \sum_i \min(1, Np_i) \leq \min(N, M).$$

Setting $M = 2^H$ gives the stated bound. Since $2^H \leq 2^H \log N$ for $N \geq 2$, the theorem holds as stated with the $O(2^H \log N)$ form. □

Corollary 4 (Sublinear Expert Growth)**.** *The ratio of new experts created to total queries satisfies $\mathbb{E}[\textit{new experts}]/N \leq O(2^{H(P_{\mathcal{M}})} \log N/N) \to 0$ as $N \to \infty$. The system learns faster than it exhausts novelty: most queries are cache hits for large N.*

Theorem 8 (Online Expert Acquisition Cost)**.** *Under the any-match routing rule and a stationary distribution $P_{\mathcal{M}}$ with entropy H, let $K_N = O(2^H \log N)$ be the total number of expert creation events after N queries (Theorem 7). The total cost attributable to expert* acquisition*—the overhead beyond running the already-learned library from the start—satisfies:*

$$C_{\text{acq}}(N) \;=\; K_N \cdot N_{\min} \cdot C_{\text{full}} \;=\; O\big(N_{\min} \cdot 2^H \log N\big) \cdot C_{\text{full}},$$

since each of the K_N expert creations requires $N_{\min}$ cache misses before triggering, each at cost C_{full}. The amortized acquisition cost per query is:

$$\frac{C_{\text{acq}}(N)}{N} \;=\; O\!\left(\frac{N_{\min} \cdot 2^H \log N}{N}\right) \cdot C_{\text{full}} \;\to\; 0 \quad \textit{as } N \to \infty.$$

In the limit, expert acquisition is free on a per-query basis.

Proof. Each expert is created after $N_{\min}$ cache misses accumulate at its signpost node (Algorithm 1). By Theorem 7, $K_N \le O(2^H \log N)$ experts are created over N queries. The acquisition cost for each expert is at most $N_{\min} \cdot C_{\text{full}}$ (the misses that triggered it, each costing one full forward pass). Total acquisition cost $\le K_N \cdot N_{\min} \cdot C_{\text{full}} = O(N_{\min} \cdot 2^H \log N) \cdot C_{\text{full}}$. Dividing by N and noting $(\log N)/N \to 0$ gives the stated bound with the $N_{\min}$ factor absorbed into the $O(\cdot)$. □ □

Remark 2 (Scope of the Bound)**.** *This theorem bounds the* acquisition overhead*—the cost of the triggering misses that create new experts—not the total miss cost, which also includes misses on light-trie-nodes (probability $< 2^{-H}$) that never accumulate enough visits to trigger expert creation. The total miss rate converges to $1 - H_\infty$ asymptotically (by Theorem 6), but the rate of convergence depends on the full distribution $P_{\mathcal{M}}$ and $N_{\min}$ and is not bounded here. The theorem establishes that the one-time cost of building the expert library is sublinear in N—a necessary condition for LAWS to be economically viable at scale.*

Theorem 9 (Abort-and-Replan Threshold)**.** *Let $C_{\text{full}} > 0$ be the cost of a full base-model forward pass, $C_{\text{hit}} > 0$ the cost of an expert evaluation, and $\lambda \ge 0$ the downstream cost per unit output error. The optimal validity radius that minimizes expected total cost is:*

$$\tau^* = \frac{2(C_{\text{full}} - C_{\text{hit}})}{\lambda \cdot \|H_W\|_{\text{op}}^2 \cdot \Lambda(W)^2 \cdot C_E^2},$$

where $\|H_W\|_{\text{op}}$ is the Hessian operator norm at the signpost. At $d_{\mathcal{T}}(x, n^*) = \tau^*$, the marginal cost of using the cache equals the marginal cost of full inference.

Proof. The expected total cost when using a Level-2 expert with routing radius τ is:

$$\text{Cost}(\tau) = \underbrace{C_{\text{hit}} \cdot P(\text{hit})}_{\text{cheap path}} + \underbrace{C_{\text{full}} \cdot P(\text{miss})}_{\text{expensive path}} + \underbrace{\lambda \cdot \mathbb{E}[\text{error}^2 \mid \text{hit}]}_{\text{error cost}}.$$

Error cost model. By Theorem 4, for a hit at query x with divergence embedding $r = \|E(x_{\bar{d}+1}) - E(n^*_{\bar{d}+1})\|$, the squared error is $\le \frac{1}{4}\|H_W\|_{\text{op}}^2 \cdot r^4$. We model the expected squared r for queries routed to the expert as increasing with τ: concretely, for the local approximation $\mathbb{E}[r^2 \mid \text{hit}] \approx (\Lambda(W) C_E)^2 \cdot \tau^2$, which holds when the conditional distribution of r given $d_{\mathcal{T}}(x, n^*) \le \tau$ concentrates at $r \approx \Lambda(W) C_E \cdot \tau$ (the Lipschitz-propagated trie distance). Under this approximation:

$$\mathbb{E}[\text{error}^2 \mid \text{hit}] \;\approx\; \tfrac{1}{4}\|H_W\|_{\text{op}}^2 \cdot (\Lambda(W) C_E)^2 \cdot \tau^2.$$

Miss probability model. $P(\text{miss}) = 1 - P_{\mathcal{M}}(\mathcal{B}(n^*, \tau)) \approx 1 - c\tau$ for small τ, where $c = \lim_{\tau \to 0} P_{\mathcal{M}}(\mathcal{B}(n^*, \tau))/\tau$ is the local $P_{\mathcal{M}}$-density in trie metric space.

Optimization. Differentiating $\text{Cost}(\tau)$ and setting to zero:

$$-(C_{\text{full}} - C_{\text{hit}}) \cdot c + \tfrac{\lambda}{2}\|H_W\|_{\text{op}}^2 \cdot (\Lambda(W) C_E)^2 \cdot \tau = 0,$$

giving $\tau^* = 2(C_{\text{full}} - C_{\text{hit}}) \cdot c / (\lambda \cdot \|H_W\|_{\text{op}}^2 \Lambda(W)^2 C_E^2)$. For the canonical case $c = 1$ this simplifies to the stated formula. □

Remark 3 (Too Close for Missiles, Switching to Guns)**.** *The Abort-and-Replan Threshold theorem formalizes the fighter pilot's abort signal. When $d_{\mathcal{T}}(x, n^*) > \tau^*$, the input is "too far from the signpost"—the cached expert cannot engage. The system aborts the cache-hit path and switches to the base model ("guns"). The abort condition is a single comparison, $O(1)$, requiring no model queries. τ^* is precomputed from W at model-load time, exactly as the minimum missile engagement range is computed during mission planning.*

5.5 LAWS as a Generalization

Theorem 10 (LAWS Generalizes MoE and KV Caching)**.** *(a)* ***MoE as a special case.*** *Any MoE model with K experts and router R is equivalent to a LAWS system with K fixed Level-1 experts and a PLT trie of depth 1.*

(b) ***KV caching as a special case.*** *Standard KV prefix caching is equivalent to LAWS with $\tau^* = -\log P_{\mathcal{M}}(n^*)$ (prefix match) and $f = \text{identity}$ (return cached activations verbatim).*

(c) ***Strict containment.*** *LAWS with online expert creation is not representable as any fixed-K MoE or as exact KV caching.*

Proof. (a) Given MoE experts $\{e_1, \ldots, e_K\}$, construct a PLT trie with K leaf nodes $n_1, \ldots, n_K$ at depth 1. Assign trie edge weights $P(n_k \mid \text{root}) = \mathbb{E}_x[R(x)_k]$ (the expected routing probability of expert k). Set $f_{n_k} = e_k$ and $\tau^*(n_k) = \infty$ (each expert covers its entire Voronoi cell in the abstract expert space). Define the parameter extractor ϕ so that expert e_k is selected whenever $R(x)_k$ is largest (i.e., the MoE router outcome determines which expert is selected). Since the PLT trie here indexes expert labels rather than linguistic token prefixes, and $au^* = \infty$ ensures every query is a cache hit, LAWS inference replicates MoE top-1 routing exactly. For top-k' sparse MoE, use k' routing balls with $au^* = \infty$ and weighted averaging. ✓

(b) Standard KV prefix caching stores activations for exact token prefixes. A LAWS expert with $n^* = \text{prefix}$, $f = \text{identity}$ (return stored KV), $\tau^* = -\log P_{\mathcal{M}}(n^*)$ (matching all x for which n^* is a prefix), is exactly a KV cache entry under any-match routing. ✓

(c) For strict containment: LAWS creates experts online from cache misses, so $|\mathcal{L}|$ is unbounded and grows with N. Any fixed-K MoE has $|\mathcal{L}| = K < \infty$ fixed at training. As $N \to \infty$, $|\mathcal{L}_{\text{LAWS}}| \to \infty > K$. LAWS cannot be represented by a fixed-K MoE. The same argument applies to KV caching with a finite cache size. ✓ □

Corollary 5 (LAWS is Strictly More Expressive)**.** *For any fixed-K MoE or finite KV cache, there exists a query distribution P and a query count $N^*(P, K)$ such that for all $N > N^*(P, K)$, LAWS achieves strictly higher expected hit rate than the MoE or KV cache.*

Proof. By Theorem 6, LAWS's hit rate is non-decreasing and converges to P-coverage of the expert library. For a fixed-K MoE or cache, hit rate is bounded by $\sum_{k=1}^{K} P(\mathcal{B}(n_k^*, \tau_k^*))$, which is fixed. LAWS's coverage grows with N, eventually exceeding the fixed system's coverage for any P with entropy > 0. □

5.6 Automatic Symbolic Vocabulary

Theorem 11 (Automatic Symbolic Vocabulary)**.** *Let $\mathcal{V}_\varepsilon(\mathcal{M})$ denote the set of PLT trie nodes n with probability mass $P_{\mathcal{M}}(n) \geq \varepsilon$. $\mathcal{V}_\varepsilon(\mathcal{M})$ constitutes a* symbolic vocabulary *with the following provable properties:*

(a) ***Coverage:*** *$\mathcal{V}_\varepsilon$ contains at most $\lfloor 1/\varepsilon \rfloor$ nodes (count bound). For the probability covered by $\mathcal{V}_\varepsilon$: by Markov's inequality applied to the surprisal $-\log_2 P_{\mathcal{M}}(x)$,*

$$P_{\mathcal{M}}(\mathcal{V}_\varepsilon) = P_{\mathcal{M}}\big(P_{\mathcal{M}}(x) \geq \varepsilon\big) \;\geq\; 1 - \frac{H(P_{\mathcal{M}})}{\log_2(1/\varepsilon)},$$

where $H(P_{\mathcal{M}})$ is the Shannon entropy. In particular, for $\varepsilon = 2^{-H/\delta}$, the vocabulary $\mathcal{V}_\varepsilon$ covers at least $1-\delta$ of the probability mass.

(b) ***Compositionality:*** *for any $n, n' \in \mathcal{V}_\varepsilon$, their longest common prefix $n \wedge n'$ is also in $\mathcal{V}_\varepsilon$ whenever $P_{\mathcal{M}}(n \wedge n') \geq \varepsilon$.*

(c) ***Graded similarity:*** *the trie metric $d_{\mathcal{T}}$ is a well-defined pseudometric on $\mathcal{V}_\varepsilon$, giving graded rather than binary match/no-match.*

(d) ***Self-certification:*** *validity of any symbolic approximation using $n \in \mathcal{V}_\varepsilon$ as a signpost is certified by $\Lambda(W)$ without human authorship or additional inference.*

(e) ***Automatic discovery:*** *$\mathcal{V}_\varepsilon(\mathcal{M})$ is derived entirely from W and $P_{\mathcal{M}}$—no human intervention is required.*

Proof. (a) Let $\mathcal{F}_\varepsilon \subseteq \mathcal{V}_\varepsilon$ be the *frontier*: the maximal (antichain) elements of $\mathcal{V}_\varepsilon$, i.e., nodes with $P_{\mathcal{M}}(n) \geq \varepsilon$ that have no proper-prefix ancestor also in $\mathcal{V}_\varepsilon$. The frontier nodes are disjoint (no sequence matches two incomparable trie paths), so $\sum_{n \in \mathcal{F}_\varepsilon} P_{\mathcal{M}}(n) \leq 1$, giving $|\mathcal{F}_\varepsilon| \leq \lfloor 1/\varepsilon \rfloor$. The full $\mathcal{V}_\varepsilon$ may be larger (containing all ancestors of frontier nodes), but the count bound applies to the frontier.

For the coverage bound: by Markov's inequality applied to the random variable $-\log_2 P_{\mathcal{M}}(x)$ (the surprisal), whose expectation is $H(P_{\mathcal{M}})$:

$$P_{\mathcal{M}}\big(-\log_2 P_{\mathcal{M}}(x) \geq \log_2(1/\varepsilon)\big) \;\leq\; \frac{H(P_{\mathcal{M}})}{\log_2(1/\varepsilon)}.$$

The complementary event $-\log_2 P_{\mathcal{M}}(x) < \log_2(1/\varepsilon)$ is exactly $P_{\mathcal{M}}(x) > \varepsilon$, i.e., $x \in \mathcal{V}_\varepsilon$. Therefore $P_{\mathcal{M}}(\mathcal{V}_\varepsilon) \geq 1 - H/\log_2(1/\varepsilon)$. Setting $\varepsilon = 2^{-H/\delta}$ gives coverage $\geq 1-\delta$. ✓

(b) The PLT trie is a prefix tree: every prefix of a sequence in the trie is also in the trie. If $n \wedge n'$ has $P_{\mathcal{M}} \geq \varepsilon$, it is by definition in $\mathcal{V}_\varepsilon$.

(c) The trie metric satisfies: $d(s,s) = -\log P(s) \geq 0$, $d(s,s') = d(s',s)$ (symmetry), and the ultrametric inequality $d(s,s'') \leq \max(d(s,s'), d(s',s''))$; see [15]. Since $d(s,s) = 0$ only when $P(s) = 1$ (a degenerate case), $d_{\mathcal{T}}$ is technically a *pseudoultrametric* rather than a true metric, giving graded rather than binary similarity.

(d) Theorem 3.

(e) The construction of $\mathcal{V}_\varepsilon$ uses only W (to compute $P_{\mathcal{M}}$ and run inference to collect samples) and the PLT trie structure, both derived from W without human authorship. □

6 Parametrized Experts: Construction and Theory

6.1 Pattern Recognition

Given a trie node n^* with $N_{\min}$ samples, the *pattern recognition* step fits the cheapest function from a candidate class hierarchy.

Proposition 1 (PAC Bounds for Pattern Recognition)**.** *For a trie node n^* with samples $\{(x_i, y_i)\}_{i=1}^N$ drawn from the subtree distribution, and a hypothesis class $\mathcal{F}$ with fat-shattering dimension d_{fat}, the pattern recognition step selects $f \in \mathcal{F}$ satisfying $\|y - f(\phi(x))\| \leq \varepsilon_{\text{fit}}$ for all samples. With probability $\geq 1 - \delta_{\text{PAC}}$, this f also satisfies the bound on* unseen *inputs in the subtree, given $N \geq N_{\min}$ where:*

$$N_{\min}(\varepsilon_{\text{fit}}, \delta_{\text{PAC}}, d_{\text{fat}}) = O\left(\frac{d_{\text{fat}}}{\varepsilon_{\text{fit}}^2} \log \frac{d_{\text{fat}}}{\delta_{\text{PAC}}}\right).$$

For primitive classes: linear ($d_{\text{fat}} = O(k \cdot d_{\text{out}})$, where k is the parameter dimension), lookup ($d_{\text{fat}} = O(|\text{table}|)$), and template ($d_{\text{fat}} = O(\textit{template length})$).

Proof. Standard uniform convergence bound for real-valued function classes (Bartlett and Mendelson [4]). For regression with ℓ^2 loss, the fat-shattering dimension at scale ε_{fit} governs the sample complexity. The VC dimension bounds follow from the standard results for each function class. □

6.2 Small MLP Approximation

Theorem 12 (MLP Fitting Bound)**.** *For any continuous function $f : \mathbb{R}^k \to \mathbb{R}^{d_{\text{out}}}$ on a compact domain D (the parameter space restricted to the validity ball) with Lipschitz constant $C_f \leq \Lambda(W)$, there exists a ReLU MLP with at most:*

$$N_{\text{neurons}} = O\Big(k \cdot d_{\text{out}} \cdot (C_f/\varepsilon)^k\Big)$$

neurons that approximates f uniformly to within ε on D. Here $d_{\text{out}} = |V|$ when approximating full logit output, or $d_{\text{out}} = d_{\text{model}}$ when approximating an intermediate hidden state.

Proof. By the Barron approximation theorem [1]: for functions with bounded spectral norm in k input dimensions, a single-hidden-layer network with m neurons achieves L^2 error $O(C_f^2/m)$, giving $m = O(C_f^2/\varepsilon^2)$ for L^2 approximation. For uniform (L^∞) approximation on a compact domain, the Cybenko theorem and subsequent quantitative refinements (e.g., Yarotsky [23]) give $O((C_f/\varepsilon)^k)$ neurons for depth-$O(\log(1/\varepsilon))$ networks. Multiplying by d_{out} output dimensions gives the stated bound. □

Corollary 6 (Small MLP Size for Low-Dimensional Parameters)**.** *For a trie node where k (the number of high-entropy parameter positions, per Theorem 5) is small (e.g., $k = 3$ for binary search, $k = 2$ for arithmetic), the MLP has $O(d_{\text{out}} \cdot (C_f/\varepsilon)^3)$ neurons. For $d_{\text{out}} = d_{\text{model}} = 4096$ (caching the final hidden state rather than full vocabulary logits, which is typical for chained LAWS inference), $C_f = 0.1$ (normalized), $\varepsilon = 0.01$: $N_{\text{neurons}} \approx 4096 \cdot 10^3 \approx 4 \times 10^6$. This MLP has $< 50\,MB$ footprint—evaluable in microseconds on a GPU versus milliseconds for a full forward pass.*

6.3 Cross-Architecture Portability

Theorem 13 (Cross-Architecture Expert Portability)**.** *A LAWS expert $(n^*, f, \phi, \tau^*, \varepsilon_{\text{fit}})$ constructed from base model $\mathcal{M}_1$ is valid for a different base model $\mathcal{M}_2$ if:*

$$\max_{x \in \mathcal{B}(n^*, \tau^*)} \|F_{W_2}(x) - f(\phi(x))\| \leq \delta.$$

This condition is testable by sampling N inputs from $\mathcal{B}(n^*, \tau^*)$ and checking the bound; with confidence $1 - \delta_{\text{PAC}}$, PAC bounds from Proposition 1 apply. The routing radius for $\mathcal{M}_2$ is $\tau_2^* = (\delta - \varepsilon_{\text{fit}}^{(2)} - 2\,\Lambda(W_2) \cdot C_E)/(\Lambda(W_2) \cdot C_E)$, where $\varepsilon_{\text{fit}}^{(2)} = \|F_{W_2}(n^*) - f(\phi(n^*))\|$ is the fitting error re-measured on $\mathcal{M}_2$.

Proof. The expert f represents a computation, not a specific model. Its validity for $\mathcal{M}_2$ depends only on whether F_{W_2} approximates f on the domain, independently of how f was constructed. Testing against $\mathcal{M}_2$'s outputs is an independent PAC learning problem with the same sample complexity bounds. □

7 The Cinderella Effect

7.1 The Problem

A natural question in any attention-sparsification scheme is whether a token with small attention weight at layer ℓ might become important at a later layer $\ell' > \ell$—the *Cinderella effect*: a token plucked from obscurity by a later layer's attention mechanism, its small initial weight amplified into a large influence on the final output.

Definition 5 (Cinderella Event)**.** *A* Cinderella event *at layer ℓ, position j occurs when: $a_{ij}^{(\ell)} \le \varepsilon$ (small attention weight at layer ℓ) but the counterfactual output difference from zeroing this entry satisfies: $\|\Delta\text{Output}_i^{(L)}\| > \alpha$ (large influence on final output) for some significance threshold $\alpha \gg \varepsilon$.*

7.2 The Cascade Bound

Theorem 14 (Cinderella Cascade Bound)**.** *If token j has attention weight $a_{ij}^{(\ell)} \le \varepsilon$ at layer ℓ, then:*

(a) The worst-case *cascade satisfies:*

$$\|\Delta\text{Output}_i^{(L)}\| \le \varepsilon \cdot \|\mathbf{v}_j^{(\ell)}\| \cdot \prod_{m=\ell}^{L-1} \kappa^{(m)}.$$

(b) A Cinderella event (large final impact despite small initial weight) can only occur if token j's surprisal satisfies:

$$h_j \ge h_{\text{Cin}} = \frac{d_{\text{head}} \cdot \log^2(\alpha/\varepsilon)}{4\kappa^2 C_E^2 \ln 2 \cdot \|q\|^2\|k\|^2},$$

where $\kappa = \max_\ell \kappa^{(\ell)}$ is the maximum per-layer coupling constant from Theorem 1, C_E is the embedding diameter, and d_{head} is the per-head dimension. (The $\log^2$ arises from squaring the logit-increase condition $2\kappa C_E\sqrt{h_j \ln 2}\|q\|\|k\|/\sqrt{d_{\text{head}}} \ge \log(\alpha/\varepsilon)$ when solving for h_j.)

(c) For tokens with $h_j < h_{\text{Cin}}$ (low-surprisal tokens), the Cinderella event cannot occur. These tokens are safely sparsifiable.

Proof. **Part (a).** Zeroing attention entry $a_{ij}^{(\ell)}$ introduces error $e_i^{(\ell)} = a_{ij}^{(\ell)} \cdot \mathbf{v}_j^{(\ell)}$, with $\|e_i^{(\ell)}\| \le \varepsilon\|\mathbf{v}_j^{(\ell)}\|$. This error enters the residual stream and propagates through layers $\ell+1, \ldots, L$. Each layer m is $\kappa^{(m)}$-Lipschitz (Lemma 1), so:

$$\|e_i^{(m+1)}\| \le \kappa^{(m)}\|e_i^{(m)}\|.$$

Iterating from ℓ to L: $\|\Delta\text{Output}_i\| \le \varepsilon\|\mathbf{v}_j^{(\ell)}\| \prod_{m=\ell}^{L-1} \kappa^{(m)}$. ✓

Part (b). For a Cinderella event, the attention weight of j at layer $\ell+1$ must increase substantially: $a_{ij}^{(\ell+1)} \ge \alpha \gg \varepsilon$. The logit for token j at layer $\ell+1$, relative to what it would be if token j's embedding had been replaced by its context-predicted mean (the counterfactual baseline), changes by:

$$\Delta\text{logit}_j = \frac{1}{\sqrt{d_{\text{head}}}}\big(\mathbf{q}_i^{(\ell+1)} \cdot \Delta\mathbf{k}_j^{(\ell+1)} + \Delta\mathbf{q}_i^{(\ell+1)} \cdot \mathbf{k}_j^{(\ell+1)}\big),$$

where $\Delta\mathbf{k}$ and $\Delta\mathbf{q}$ are the deviations of the key/query vectors from this baseline. By Theorem 1 and the surprisal-bounded embedding variance of [17], a token with per-token surprisal h_j has expected embedding deviation $\leq C_E\sqrt{h_j \ln 2}$ from its context-predicted mean (using the updated variance-entropy bound), which propagates through the κ-Lipschitz layer map to give:

$$\|\Delta\mathbf{k}_j^{(\ell+1)}\|, \|\Delta\mathbf{q}_i^{(\ell+1)}\| \leq \kappa \cdot C_E \cdot \sqrt{h_j \ln 2},$$

so $|\Delta\text{logit}_j| \leq 2\kappa C_E\sqrt{h_j \ln 2} \cdot \|q\|\|k\|/\sqrt{d_{\text{head}}}$. For the attention weight to increase from ε to α, the logit must increase by at least $\log(\alpha/\varepsilon)$ (to leading order, assuming other attention scores are approximately unchanged). Setting $2\kappa C_E\sqrt{h_j \ln 2}\|q\|\|k\|/\sqrt{d_{\text{head}}} \geq \log(\alpha/\varepsilon)$ and solving for h_j gives $h_j \geq h_{\text{Cin}}$.

Part (c). The contrapositive: $h_j < h_{\text{Cin}} \Rightarrow$ Cinderella event cannot occur (logit increase is too small to bring j from ε-weight to α-weight). ✓ □

Corollary 7 (Shannon Overflow Set)**.** *Define the* Shannon overflow set $\mathcal{O}_\varepsilon = \{j : h_j \geq h_{\text{Cin}}\}$. *Tokens in $\mathcal{O}_\varepsilon$ may exhibit Cinderella events and must be retained in full-precision computation. Tokens not in $\mathcal{O}_\varepsilon$ are safe to sparsify or cache-hit. The expected size of $\mathcal{O}_\varepsilon$ is:*

$$\mathbb{E}[|\mathcal{O}_\varepsilon|] = n \cdot P_{\mathcal{M}}(h_j \geq h_{\text{Cin}}) \leq n \cdot \frac{\log_2 \text{PP}(\mathcal{M})}{h_{\text{Cin}}},$$

which approaches zero as $h_{\text{Cin}} \to \infty$ (larger threshold, smaller overflow set).

Proof. Markov's inequality applied to the surprisal random variable: $P(h_j \geq h_{\text{Cin}}) \leq \mathbb{E}[h_j]/h_{\text{Cin}} = \log_2 \text{PP}(\mathcal{M})/h_{\text{Cin}}$. □

8 Robotics and Fleet Learning

8.1 The Robotics Challenge

Modern robot controllers—whether model-based (MPC, trajectory optimization) or learned (diffusion policy, transformer-based visuomotor)—face a fundamental tension: they must be accurate (avoiding catastrophic failure) yet fast (100+ Hz control loops). Full trajectory optimization or neural network inference at each control timestep is expensive. Current approaches use one of: (1) hard-coded motion primitives (inflexible, brittle to novel objects); (2) offline-trained policies (cannot generalize to new environments without retraining); or (3) online adaptation (expensive, requires backpropagation at runtime).

LAWS offers a fourth path: automatic discovery and certification of motion primitives from actual task executions, growing richer with fleet experience, downloadable as compact updates.

8.2 LAWS for Robotic Motor Programs

Definition 6 (Robotic LAWS Expert)**.** *A* robotic LAWS expert $e = (n^*_{\text{task}}, \pi, \phi, \tau^*, \varepsilon_{\text{fit}})$ *where:*

- n^*_{task} *is a PLT trie node over task descriptions (natural language or sensor embedding);*
- $\pi : \mathbb{R}^k \to \mathcal{A}^T$ *is a motor program mapping task parameters to an action sequence;*
- ϕ *extracts task parameters (object pose, target location, mass, friction coefficient);*
- τ^* *is the validity radius derived from the controller's Lipschitz constant $\Lambda(W_{\text{ctrl}})$.*

8.3 State of the Art and LAWS Improvements

Current state-of-the-art robotic policies include: *Diffusion Policy* [6] (denoising diffusion for visuomotor control), *RT-2* [5] (vision-language-action models), and π_0 [3] (flow-matching generalist policies). These achieve impressive generalization but require full neural network inference at every control step: typically 50–200 ms per action, limiting real-time control frequency.

Proposition 2 (LAWS Speedup for Repetitive Tasks)**.** *For a task distribution P_{task} with entropy H_{task} and a LAWS expert library with hit rate H_n, the expected control step latency is:*

$$\mathbb{E}[T_{\text{LAWS}}] = H_n \cdot T_{\text{expert}} + (1 - H_n) \cdot T_{\text{full}},$$

where $T_{\text{expert}} \ll T_{\text{full}}$. For a robot performing repetitive household tasks (object manipulation, item retrieval, fixture interaction) with $H_n \geq 0.9$ (achievable after sufficient deployment), $\mathbb{E}[T_{\text{LAWS}}] \approx T_{\text{expert}}$, enabling order-of-magnitude latency reduction.

8.4 Fleet Learning Theorems

Theorem 15 (Fleet Learning Lower Bound)**.** *Consider K LAWS-equipped robotic units, each performing M tasks per day, all contributing observations to a shared central LAWS library. Under a stationary task distribution P_{task}:*

(a) After D deployment days, the shared library has at most $O(2^{H(P_{\text{task}})} \log(KMD))$ experts.

(b) The fleet hit rate $H_{K,D}$ is non-decreasing in D (by Theorem 6) and converges to full coverage as $KMD \to \infty$: since $O(2^H \log(KMD))$ distinct heavy trie nodes are visited and each expert creation strictly increases the hit rate, $H_{K,D} \to 1$ for any distribution with finite entropy H.

(c) The fleet achieves the same hit rate as a single unit in time $D_K \approx D_1/K$, i.e., convergence is $\Omega(K)$ faster than single-unit deployment.

Proof. **Part (a).** By Theorem 7, each unit alone creates $O(2^H \log(MD))$ experts. With K units sharing a library, total queries are KMD. Applying Theorem 7 with $N = KMD$: $O(2^H \log(KMD))$ experts (valid when $KMD \leq N_{\min} \cdot 2^H$; for larger KMD all heavy nodes are covered and $H_{K,D} = 1$, satisfying the bound since $\min(1, \cdot) = 1$), where $H = H(P_{\text{task}})$. ✓

Part (b). By Theorem 6, each expert creation strictly increases $H_{K,D}$. By part (a), $O(2^H \log(KMD))$ experts are created, covering an increasing fraction of the input distribution. Since $H(P_{\text{task}})$ is finite, all $O(2^H)$ heavy trie nodes are eventually visited, and $H_{K,D} \to 1$ as $KMD \to \infty$. ✓

Part (c). From part (b), both a single unit and the K-unit fleet converge to hit rate $H_{K,D} \to 1$. The fleet converges K times faster because it generates KM queries per day (versus M for a single unit), discovering the same $O(2^H)$ heavy trie nodes in $1/K$ the time. The improvement factor is $D_1/D_K = K$, giving the stated $\Omega(K)$ convergence speedup. ✓ □

Corollary 8 (Network Effect)**.** *The fleet learning benefit is superlinear in K for small K and approaches K-linear for large K (the log correction vanishes as $M \to \infty$). This formalizes the* network effect of experience*: each additional unit contributes its workload to the shared library, benefiting all other units. A fleet of 1,000 robots converges to high hit rate roughly 1,000 times faster than a single robot.*

8.5 Over-the-Air (OTA) Expert Updates

Theorem 16 (OTA Update Bandwidth Bound)**.** *The incremental LAWS expert library update covering the period $[t, t+\Delta t]$ (representing $\Delta N = KM\Delta t$ new observations) has description length:*

$$\mathcal{L}_{\Delta t} = O\Big(2^{H(P_{\text{task}})} \cdot \log(\Delta N) \cdot B_{\text{expert}}\Big) \text{ bits},$$

where B_{expert} is the description length of a single expert (Theorem 17 below). For $\Delta t = 24$ hours, $K = 1000$ units, $M = 100$ tasks/unit/day ($\Delta N = 10^5$):

$$\mathcal{L}_{\Delta t} = O\big(2^H \cdot \log(10^5) \cdot B_{\text{expert}}\big) \approx O(2^H \cdot 17 \cdot B_{\text{expert}}).$$

For $B_{\text{expert}} \approx 50\,KB$ (small MLP), $H \approx 10$ bits ($2^H = 1024$): $\mathcal{L}_{24\text{h}} \approx 870\,MB$ per day for the full fleet update. Individual robot updates are $\approx 870\,KB$ per day per unit—feasible on any connected device.

Proof. The number of new experts in period Δt is $O(2^H \log(\Delta N))$ by Theorem 7. Each expert has description length B_{expert}. Total bits: $O(2^H \log(\Delta N) \cdot B_{\text{expert}})$. Substituting $\Delta N = KM\Delta t$ gives the bound. Per-unit update: if units have approximately disjoint workload domains (each unit specializes in different task types), then on average $1/K$ of the new experts are relevant to any one unit, giving per-unit download $O(2^H \log(\Delta N) \cdot B_{\text{expert}}/K)$ bits. In the homogeneous case (all units share the same task distribution), each unit downloads all new experts and the per-unit cost equals the fleet total. □

Theorem 17 (Expert Description Length)**.** *For a trie node n^* with k parameter dimensions and validity radius τ^*, the minimum description length of a LAWS expert achieving output error $\leq \varepsilon$ is:*

$$B(n^*, \varepsilon) = \Omega\left(k \cdot d_{\text{out}} \cdot \log \frac{\tau^*}{\varepsilon}\right) \text{ bits.}$$

A small MLP expert (Theorem 12) requires $O(k \cdot d_{\text{out}} \cdot (C_f/\varepsilon)^k \cdot 32)$ bits (at fp32), which is exponentially larger in k than this lower bound. The gap reflects the curse of dimensionality *for general Lipschitz approximation; for specific structured function classes (linear, lookup, template), the description length matches the lower bound to within polylogarithmic factors.*

Proof. Lower bound: to represent a Lipschitz function $f : \mathbb{R}^k \to \mathbb{R}^{d_{\text{out}}}$ to uniform accuracy ε over a parameter ball of radius τ^*, one must distinguish $\Omega((\tau^*/\varepsilon)^k)$ cells in the k-dimensional input space (by a standard covering argument) and specify d_{out} output values per cell to precision ε. The total description length is therefore $\Omega((\tau^*/\varepsilon)^k \cdot d_{\text{out}} \cdot \log(1/\varepsilon))$ bits.

The stated bound $B(n^*, \varepsilon) = \Omega(k \cdot d_{\text{out}} \cdot \log(\tau^*/\varepsilon))$ is a weaker lower bound obtained by taking the log of the cell count: $\log((\tau^*/\varepsilon)^k) = k \log(\tau^*/\varepsilon)$, so any description of the cell index alone requires $\Omega(k \log(\tau^*/\varepsilon))$ bits, giving $\Omega(k \cdot d_{\text{out}} \cdot \log(\tau^*/\varepsilon))$ bits total for d_{out} output dimensions. The MLP construction in Theorem 12 achieves $O(k \cdot d_{\text{out}} \cdot (C_f/\varepsilon)^k) \cdot 32$ bits (at fp32); the gap between this upper bound and the lower bound reflects the exponential dependence on k inherent in approximating k-dimensional Lipschitz functions. □

Remark 4 (Differential Privacy)**.** *The OTA update can incorporate differential privacy [9] by adding calibrated Gaussian noise to the expert parameters before uploading. The privacy cost is bounded by the description length: an expert update of B bits can achieve $(\varepsilon_{\text{DP}}, \delta_{\text{DP}})$-differential privacy with noise scale $\sigma = \sqrt{2B \ln(1.25/\delta_{\text{DP}})}/\varepsilon_{\text{DP}}$. For $B = 50\,KB$, $\delta = 10^{-5}$, $\varepsilon_{\text{DP}} = 1$: $\sigma \approx 0.03$, adding 3% noise to expert parameters—within the fitting error tolerance for most experts.*

9 Comparison to Prior Work

9.1 LAWS vs. KV Caching

Standard transformer KV caching [18, 13] stores key-value vectors for exact token prefixes, enabling reuse across requests sharing a common prefix. The companion paper [17] extended this to *approximate* prefix matching via the PLT trie metric, and proved that sequential KV compression can exceed TurboQuant's per-vector Shannon limit by exploiting the sequential structure of token streams.

LAWS extends both in several fundamental ways:

- LAWS is not restricted to KV vectors—it caches *any* intermediate computation or final output.
- LAWS experts are *parametrized*: they compute a function of the input's variable parameters, not a fixed stored value.
- LAWS validity domains are formally certified by $\Lambda(W)$; standard KV caching has no validity theory.
- LAWS's library grows with deployment; KV caches have finite size with eviction policies.

Formally, Theorem 10(b) shows KV caching is the degenerate case $\tau^* = 0$, $f = \text{identity}$. LAWS with $\tau^* > 0$ and nontrivial f is strictly more powerful.

9.2 LAWS vs. Mixture of Experts

Standard MoE [20, 10, 8] maintains a fixed pool of K expert sub-networks, trained jointly with the router. Fast Feed-Forward Networks (FFF) [2] replace the router with a binary tree, achieving $O(\log K)$ routing. Symbolic MoE [21] routes to skill-based expert models selected by an LLM router.

The key distinctions from LAWS:

- **Fixed vs. growing:** all MoE variants fix K at training. LAWS's $|\mathcal{L}|$ grows unboundedly with deployment.
- **Training vs. inference-time:** MoE experts are trained jointly with the base model; LAWS experts are created at inference time from observed outputs.
- **No validity guarantee:** no MoE variant provides a formal guarantee that any expert is correct on any given input. LAWS provides $\|\text{LAWS}(x) - F_W(x)\| \leq \delta$ for all inputs in certified validity domains.
- **Routing basis:** MoE routing is learned; LAWS routing uses the PLT trie metric derived from W, requiring no routing training.

Theorem 10(a) shows MoE is the LAWS special case with fixed $\mathcal{L}$ and depth-1 trie.

9.3 LAWS vs. Cyc

Cyc [14] encoded common-sense knowledge as explicit logical axioms, requiring approximately 47,000 person-years of manual knowledge entry. Its knowledge base is static, brittle (fails outside authored rules), and provides no formal guarantee of correctness.

LAWS shares Cyc's goal (a reusable knowledge library for fast symbolic-style inference) but differs fundamentally: its vocabulary is discovered automatically from the training distribution

(Theorem 11), grows with deployment, provides formal validity certificates, and requires no human authorship. The comparison:

Property	Cyc	LAWS
Knowledge source	Human-authored	Automatic from W
Update mechanism	Manual	Online from queries
Validity guarantee	None	$\Lambda(W)$-certified
Graceful degradation	No	Yes ($\varepsilon_{\text{fit}} + 2\Lambda C_E$ uniform bound)
Generalization beyond rules	None	Jacobian correction
Human labor required	$\gg 10^4$ person-years	None

9.4 LAWS vs. Wolfram Alpha

Wolfram Alpha [22] curates computational knowledge in specific domains (mathematics, science, geography) and provides exact symbolic computation within those domains. Outside the curated domain, it fails entirely.

LAWS differs in: (1) domain coverage is automatic (whatever the base model learned); (2) experts in Level-3 (primitive function) are exactly the kind of computation Wolfram Alpha performs—but discovered automatically; (3) generalization beyond exact matches is formal (Jacobian correction); (4) Wolfram Alpha cannot grow its knowledge base from user queries.

The deepest distinction: Wolfram Alpha's knowledge is organized by human experts according to mathematical structure. LAWS's knowledge is organized by probability structure—the trie metric $d_{\mathcal{T}}$ is the natural metric for a system that learned from language, not the human-designed ontology that organizes Wolfram's knowledge base.

10 Conjectures and Open Problems

Conjecture 1 (Effective Lipschitz Concentration)**.** *For a transformer F_W trained to near-zero cross-entropy loss on a corpus with entropy H, the* effective *Lipschitz constant on inputs drawn from $P_{\mathcal{M}}$ concentrates well below the worst-case $\Lambda(W)$:*

$$\Lambda_{\text{eff}} \;=\; \mathbb{E}_{x\sim P_{\mathcal{M}}}\left[\sup_{x':\|E(x')-E(x)\|\le\varepsilon} \frac{\|F_W(x')-F_W(x)\|}{\|E(x')-E(x)\|}\right] \;\ll\; \Lambda(W).$$

Specifically, $\Lambda_{\text{eff}} = O(\text{poly}(H))$, meaning the effective Lipschitz constant grows at most polynomially in the training entropy, while $\Lambda(W)$ can grow exponentially in L.

Evidence and partial proof sketch. *Three lines of evidence support this. First, empirically: LLMs generalize well to semantically similar inputs (paraphrases, minor reformulations), which would be impossible if the effective Lipschitz were truly exponential. Second, theoretically: for a model minimizing the cross-entropy $H(P_{\mathcal{M}})$, the output $F_W(x)$ approximates the probability vector $P_{\mathcal{M}}(\cdot \mid x)$. Small perturbations to x that preserve the conditional distribution $P_{\mathcal{M}}(\cdot \mid x)$ produce near-zero output change. The fraction of perturbations that meaningfully change $P_{\mathcal{M}}(\cdot \mid x)$ is bounded by the local entropy $H(t_i \mid t_{<i})$, which is small for coherent text. Formally, if $\|P_{\mathcal{M}}(\cdot \mid x) - P_{\mathcal{M}}(\cdot \mid x')\|_1 \le \delta$ whenever $\|E(x_{\bar{d}+1}) - E(x'_{\bar{d}+1})\| \le \varepsilon$, then $\|F_W(x) - F_W(x')\| \le \delta$ (via the softmax Lipschitz bound), so $\Lambda_{\text{eff}} \le \delta/\varepsilon$ on the distribution. Third: the Theorem 2 lower bound $\Lambda(W) \ge \Delta/r_{\min}$ is tight for high-precision tasks but loose for typical language tasks where Δ is small (consecutive tokens produce similar distributions). Proving this rigorously would require tight bounds on $\|P_{\mathcal{M}}(\cdot \mid x) - P_{\mathcal{M}}(\cdot \mid x')\|_1$*

as a function of the embedding perturbation, which depends on the model's spectral properties near the training distribution—an open problem.

Conjecture 2 (Phase Transition in LAWS Convergence)**.** *Under a stationary distribution $P_{\mathcal{M}}$ with entropy H, the LAWS hit rate H_N as a function of total queries N exhibits a* sharp phase transition*: there exists $N^* = \Theta(N_{\min} \cdot 2^H)$ such that $H_N \approx 0$ for $N \ll N^*$ and $H_N \approx H_\infty$ for $N \gg N^*$, with the transition width $O(N^*/\sqrt{H})$.*

Sketch. *This mirrors the coupon-collector phase transition [4]: collecting K coupons with equal probability $1/K$ shows a sharp transition at $N = K \ln K$ with width $O(K)$. For LAWS with $K = 2^H$ heavy nodes and per-node visit threshold $N_{\min}$: the last heavy node is covered at $N \approx N_{\min} \cdot 2^H \ln(2^H) = N_{\min} \cdot H \cdot 2^H$, and the transition sharpness is $O(N_{\min} \cdot 2^H)$. The non-uniform case (expert nodes have different probabilities) softens the transition but preserves the qualitative structure—the heaviest experts are covered early, then a long "tail" period covers rare heavy nodes.*

A formal proof would follow from applying the Erdős–R'enyi coupon-collector result to the trie heavy node occupancy process. The main technical difficulty is that LAWS experts are not independent: covering trie node n^* may partially cover subtree descendants. The transition is therefore faster *than the standard coupon collector (positive correlation across experts accelerates coverage), so the phase transition is at most $N^* = O(N_{\min} \cdot 2^H \cdot H)$.*

Conjecture 3 (Symbolic Pattern Emergence)**.** *For a sufficiently capable base model $\mathcal{M}$ trained on a corpus containing code, mathematics, and structured data, all PLT trie nodes n^* with $P_{\mathcal{M}}(n^*) \geq \varepsilon$ have experts in one of a finite set of primitive function classes (linear, lookup, arithmetic, template, small MLP) with probability approaching 1 as $\varepsilon \to 0$.*

Sketch: *High-probability patterns are those the model has seen many times. Gradient descent over many presentations drives the model toward the minimum-description-length function consistent with the pattern—which for algorithmic patterns (sorting, arithmetic, lookup) is the primitive function itself. Formally: if f^* is the MDL function fitting the samples at n^*, and the model F_W approximately achieves MDL on in-distribution data (by the PAC-Bayes bound [4]), then $F_W \approx f^*$ uniformly over the subtree of n^*. For the primitive classes the model's implicit regularization selects (documented empirically by grokking [21]), this means f^** is *a primitive function. Proof requires a characterization of MDL functions representable by transformers, currently open.*

Conjecture 4 (Optimal Chunking at Surprisal Peaks)**.** *The optimal hierarchical decomposition of a LAWS expert library—minimizing total description length under a block-decomposition constraint—places chunk boundaries at positions i of locally maximal conditional entropy $H_i = H(t_i \mid t_{<i})$.*

Sketch and partial proof. *At any position i, the gain from introducing a chunk boundary is bounded by the mutual information $I(t_i; \text{expert label} \mid t_{<i})$, which is maximized when H_i is maximized (because high-entropy positions are where the model's output is most sensitive to the choice of t_i, and thus where a new signpost reduces approximation error most). More precisely: the description length of the trie decomposes as $L(\mathcal{T}) = \sum_i H(n_i^* \mid t_{<i})$, and the greedy algorithm that splits at the position $i^* = \arg\max_i H_i$ is equivalent to Huffman coding on the surprisal sequence. The greedy algorithm is provably optimal for binary block decomposition [12]; the generalization to k-ary splits requires the equivalent of the optimality of Huffman codes for k-ary alphabets.*

Conjecture 5 (Cross-Domain Transfer via Shared Trie)**.** *For two models $\mathcal{M}_1$ (language) and $\mathcal{M}_2$ (robotics) with a shared natural-language task representation, LAWS experts constructed from $\mathcal{M}_1$'s language outputs transfer to $\mathcal{M}_2$'s robot actions for tasks within the shared trie node's subtree, with validity certified by $\Lambda(W_2)$ applied to the transferred expert.*

Sketch. *The shared PLT trie indexes tasks by their linguistic description. If $\mathcal{M}_1$ and $\mathcal{M}_2$ both receive the same natural-language description as input and produce semantically aligned outputs (as*

in vision-language-action models [5, 3]), then the PLT node n^* identifying the task is common to both models. The LAWS expert $e = (n^*, f_1, \phi, \tau^*)$ built from $\mathcal{M}_1$'s outputs can be re-certified for $\mathcal{M}_2$ by evaluating $\varepsilon_{\text{fit}}^{(2)} = \|F_{W_2}(n^*) - f_1(\phi(n^*))\|$ on a small validation set (Theorem 13). The key empirical claim—that $\varepsilon_{\text{fit}}^{(2)}$ is small whenever the models share a task representation—is supported by the grokking literature but requires formal verification.

Conjecture 6 (LAWS Convergence Rate Lower Bound)**.** *No online inference caching algorithm can achieve acquisition cost $o(2^{H(P_{\mathcal{M}})} \cdot \log N)$ expert creations in the worst case over stationary distributions with entropy H.*

Sketch. *This is an information-theoretic lower bound: to achieve hit rate H_∞ on a distribution with 2^H equally-probable heavy nodes, any algorithm must "discover" each heavy node at least once. In the online setting (queries arrive sequentially, no lookahead), the first discovery of node n_k^* requires at least one query that hits n_k^*. The expected number of queries to first hit all 2^H nodes is $\Omega(2^H \log 2^H) = \Omega(2^H \cdot H)$ by the standard coupon collector lower bound. Divided by the $N_{\min}$ threshold, this gives $\Omega(2^H \cdot H/N_{\min})$ expert creations. Since $H \leq \log_2 N$ (by the AEP), this matches the upper bound $O(2^H \log N)$ of Theorem 7 to within constant factors, establishing that LAWS is* acquisition-optimal *among stationary online caching algorithms. Formal proof requires a coupon-collector lower bound argument in the online setting, which is standard but involves careful handling of adaptive query distributions.*

11 Discussion

11.1 LAWS for Diffusion Models

In diffusion model inference, the base computation maps $(x_t, t, c) \to x_{t-1}$ where x_t is the noisy image, t is the timestep, and c is the conditioning. This is repeated 20–50 times per generation.

A LAWS expert for diffusion covers a trie node in the conditioning space c for a timestep range $[t_1, t_2]$. Early denoising steps (high noise, low specificity) are well-suited to LAWS: the conditioning c largely determines the coarse structure of the output, which is highly predictable for common prompts. Expert functions at these steps can be linear or low-rank (the denoising direction is approximately the same for nearby prompts). Late denoising steps (fine detail) are less suitable—the output is sensitive to exact conditioning, and validity radii are small.

The practical implication: for repeated or similar prompts (style transfer, batch generation with variations), LAWS eliminates the early denoising steps, replacing them with cheap expert evaluations. At 20 denoising steps with 10 early-step hits per generation, LAWS reduces inference cost by 50%.

11.2 Hardware Implementation

The LAWS architecture maps naturally to a hardware design: an *Attention Signpost Cache* (ASC) processor, analogous to an L2 cache but for neural activations. Key hardware components:

- **Trie index unit:** fast hash-addressed lookup of trie node IDs, $O(1)$ per query after PLT construction.
- **Weight-norm compute unit:** computes $\Lambda(W)$ and Jacobians at model load time; a one-time $O(Ld_{\text{model}}^2)$ cost.
- **Expert SRAM:** on-die storage for expert parameters (Jacobians, MLP weights), ∼500 MB for a 70B model's signpost library.

- **Delta comparator:** $O(1)$ circuit comparing query distance to validity radius.
- **Jacobian multiplier:** dense matrix-vector multiply for Level-2 correction, $O(d_{\text{model}}^2)$ operations.

This architecture does not exist in current GPU designs (NVIDIA H100, AMD MI300X). The closest analog is the L2 cache in CPUs, but optimized for transformer activation patterns rather than memory addresses. A custom ASIC implementing the ASC could reduce inference latency for in-distribution queries by 10× or more.

11.3 LAWS as an AI Architecture Paradigm

LAWS represents a new paradigm for AI inference that we term *Learning from Actual Workloads Symbolically*: the combination of a powerful but expensive base model with a self-certifying, self-growing library of cheap symbolic approximations discovered automatically from deployment.

This paradigm has antecedents in computer architecture (L1/L2/L3 caches), in cognitive science (System 1/System 2, motor chunking, expertise), and in linguistics (Chomsky's innate prior, parameter setting in UG). What is new is the formal certification: unlike all prior work, LAWS provides δ-accuracy guarantees for every expert, derived from the model's own weights.

We believe LAWS is the correct architecture for the era of deployed AI. As LLMs, robotic controllers, and diffusion models become commodities running on billions of devices, the question is not whether to cache—it is how to certify that caching is safe. LAWS answers this question.

11.4 Limitations

The main limitation of this work is that the Lipschitz constant $\Lambda(W)$ can be large for deep networks, making validity radii very small for large errors δ. In practice, the *effective* Lipschitz constant (measured empirically on in-distribution inputs) is much smaller than the theoretical worst-case bound; this gap between worst-case and typical-case behavior is an important empirical question.

A second limitation is that Theorem 7 assumes stationarity. Real workload distributions shift over time (new tasks, new environments, new users). LAWS handles distribution shift through the cache-miss path, which creates new experts as needed; but the convergence guarantees of Theorem 15 apply only under stationarity.

12 Discovering Laws: The Scientific Analogy

12.1 Scientists Discover Laws; They Do Not Legislate Them

The name LAWS is chosen deliberately. When Newton observed the motion of planets and falling apples, he did not *invent* the law of gravitation—he *discovered* an invariant pattern latent in the empirical data. When Mendel studied peas, he discovered inheritance ratios that held across thousands of crosses. The scientific method is, at its core, a procedure for extracting cheap predictive models from expensive observations: run the experiment once (or a few times), identify the invariant pattern, and use it to predict future experiments without running them again.

LAWS formalizes precisely this operation for neural network inference. A trained model F_W encodes, in its weights, everything the training process "observed" about the distribution $P_{\mathcal{M}}$. The LAWS framework extracts the invariant patterns latent in this learned behavior—the regularities that hold across families of similar inputs—and encodes them as cheap certified experts. Future

queries matching a known pattern do not require running the full model again; they are answered by the discovered law.

The analogy extends to the dynamics of discovery. Scientists accumulate laws over time: each new experiment either confirms an existing law (cache hit) or reveals an anomaly that prompts a new theory (cache miss, new expert creation). The body of scientific knowledge—like the LAWS expert library—grows sublinearly in the number of experiments performed, because most experiments confirm known laws. Theorem 7 is the formal statement of this intuition: new laws are discovered at rate $O(2^H \log N)$, not $O(N)$.

12.2 Animals, Experts, and the System 1 Library

This pattern of law-discovery appears throughout biological intelligence. A hawk hunting prey does not recompute flight dynamics from first principles on each wingbeat; it executes motor programs refined over thousands of hunts, correcting for local wind and prey trajectory as small deltas from the cached plan. A chess grandmaster does not search the game tree to depth 20; they recognize the position as an instance of a known pattern and execute the associated strategy, engaging slow deliberate search only when the position is genuinely novel. A physician recognizes a disease presentation from a symptom constellation—"I've seen this before"—and reserves systematic diagnostic protocols for cases that don't fit any known pattern.

Kahneman [11] formalized this as dual-process cognition. *System 1* is the library of cached laws—fast, automatic, pattern-based, operating below conscious awareness. *System 2* is the base model—slow, deliberate, effortful, invoked when System 1 cannot certify an answer. The key insight Kahneman identified, and which LAWS formalizes, is that *expertise consists of System 1 richness, not System 2 speed*: the grandmaster is not faster at searching; they have a larger, more accurate library of cached patterns.

Chomsky [7] identified a related structure in language acquisition. Children acquire complex grammar from impoverished input because they bring an *innate prior*—a Language Acquisition Device—that constrains the space of possible grammars. The child does not learn grammar from scratch; they set parameters of a pre-existing structural template. In LAWS terms: the model's pre-trained weights W are the innate prior. The PLT trie derived from W is the innate expert library. Deployment queries calibrate this prior, just as linguistic exposure calibrates the LAD.

12.3 Robotics and Vehicular Workloads as a Proving Ground

Nowhere is this law-discovery process more transparent than in robotic and vehicular deployment. A domestic robot performing pick-and-place tasks across thousands of households encounters the same objects in the same functional configurations, with variation only in exact pose, lighting, and surface texture. Each successful grasp is an "observation" that can be distilled into a cheap expert: a parametrized motor program certified to work on objects within a validity ball of the observed configuration.

As the fleet scales, the laws become richer. Ten thousand robots performing household tasks are collectively running 10,000 experiments per hour. The LAWS framework provides the mechanism by which this distributed experimentation yields shared certified knowledge: each unit uploads its cache-miss observations (the anomalies), the LAWS system extracts new experts (the new laws), and all units download the update. The law of grasping cylindrical objects, the law of opening lever-handle doors, the law of navigating narrow corridors—these emerge automatically from actual workloads, not from human-programmed motion libraries.

13 Energy Savings and Edge Deployment

13.1 The Energy Cost of Neural Inference

A forward pass through a transformer with L layers, context length n, model dimension d, and H_{head} attention heads requires approximately:

$$C_{\text{full}} \approx 2 \cdot L \cdot n^2 \cdot d_{\text{model}} + 2 \cdot L \cdot n \cdot d_{\text{model}}^2 \quad \text{floating-point operations (FLOPs).}$$

For a 70B model ($L = 80$, $d = 8192$, $n = 4096$): $C_{\text{full}} \approx 7 \times 10^{13}$ FLOPs per forward pass. At the energy efficiency of an NVIDIA H100 ($\sim 2 \times 10^{15}$ FLOPs/Joule), a single forward pass consumes ~ 35 mJ. At 100 queries/second, inference for a single deployed model requires ~ 3.5 W—a significant power budget for edge devices.

13.2 Energy Cost of LAWS Cache Hits

A LAWS cache hit at a Level-2 (Jacobian) expert with k-dimensional parameter space costs:

$$C_{\text{hit}} = \underbrace{O(n)}_{\text{trie lookup}} + \underbrace{O(k)}_{\text{param extract}} + \underbrace{O(k \cdot d_{\text{model}})}_{\text{Jacobian apply}} = O(n + k\, d_{\text{model}}) \quad \text{FLOPs.}$$

For $k = 10$ (typical parameter count), $d_{\text{model}} = 8192$, $n = 4096$: $C_{\text{hit}} \approx 86{,}000$ FLOPs—a factor of $\sim 7.7 \times 10^8$ over the full forward pass.

Theorem 18 (Energy Savings Bound)**.** *Let $H_\infty = \lim_{N\to\infty} H_N$ be the asymptotic LAWS hit rate for a stationary distribution $P_{\mathcal{M}}$. The asymptotic energy consumption per query, relative to full inference, satisfies:*

$$\frac{E_{\text{LAWS}}}{E_{\text{full}}} = H_\infty \cdot \frac{C_{\text{hit}}}{C_{\text{full}}} + (1 - H_\infty) \leq 1 - H_\infty \cdot \left(1 - \frac{n + k\, d_{\text{model}}}{Ln^2 + Lnd_{\text{model}}}\right).$$

For $H_\infty = 0.9$, $k = 10$, $L = 80$, $n = 4096$, $d_{\text{model}} = 8192$:

$$\frac{E_{\text{LAWS}}}{E_{\text{full}}} \approx 0.1 + 0.9 \times 1.3 \times 10^{-9} \approx 10\%,$$

a 10× energy reduction. For smaller models on edge hardware ($L = 12$, $n = 512$, $d_{\text{model}} = 768$, as in on-device assistants), the ratio is $\approx 0.1 + 0.9 \times 6 \times 10^{-7} \approx 10\%$ with similar savings.

Proof. Expected energy per query under LAWS: $E_{\text{LAWS}} = H_\infty \cdot C_{\text{hit}} \cdot e + (1 - H_\infty) \cdot C_{\text{full}} \cdot e$ where e is energy per FLOP (hardware-dependent, cancels in the ratio). Dividing by $E_{\text{full}} = C_{\text{full}} \cdot e$ and simplifying: $E_{\text{LAWS}}/E_{\text{full}} = H_\infty \cdot C_{\text{hit}}/C_{\text{full}} + (1 - H_\infty)$. Substituting $C_{\text{hit}} = n + k\, d_{\text{model}}$ and $C_{\text{full}} = 2L(n^2 d_{\text{model}} + nd_{\text{model}}^2)$ (dominant terms) gives the bound. The numerical evaluation substitutes the stated parameter values. □

Corollary 9 (Battery Life on Edge Devices)**.** *For a mobile device running continuous LLM inference at 10 queries/second using a small on-device model ($L = 12$, $n = 512$, $d_{\text{model}} = 768$), consuming $\approx$ 0.7 mJ/query at mobile NPU efficiency (e.g., Apple Neural Engine $\sim$15 TOPS at $\sim$0.5 W; comparable energy-efficiency to an H100 per FLOP):*

- *Without LAWS: $10 \times 0.7\,mJ = 7\,mW$ average; 50 Wh / 0.007 W $\approx$ 7,000 hours continuous inference battery.*
- *With LAWS at 90% hit rate: power drops by $\approx 10\times$ to $\approx 0.7\,mW$, extending battery life proportionally to $\approx$ 70,000 hours for continuous inference.*

The 10× figure matches Theorem 18 for the given parameters.

13.3 On-Demand Expert Download

A key property of LAWS for edge deployment is *domain-selective loading*: a device need not download the entire expert library. It downloads only the experts relevant to its anticipated workload.

Definition 7 (Expert Demand Profile)**.** *The* demand profile *of a device is the set of PLT trie nodes $\mathcal{D} \subseteq \mathcal{T}(\mathcal{M})$ covering the anticipated workload with probability $\geq 1 - \delta_{\text{miss}}$:*

$$\mathcal{D} = \arg\min_{S \subseteq \mathcal{T}} |S| \quad s.t. \quad \sum_{n \in S} P_{\mathcal{M}}(n) \geq 1 - \delta_{\text{miss}}.$$

A device downloads only $\{e_n : n \in \mathcal{D}\}$, achieving hit rate $\geq 1 - \delta_{\text{miss}}$ with minimum bandwidth.

Proposition 3 (Demand-Selective Download Efficiency)**.** *For a Zipf(s) task distribution (common in practice [24]), the top-M experts cover fraction $1 - O(M^{1-s})$ of the probability mass. For $s = 1.5$ (typical web workload), the top $M = 1000$ experts cover $> 95\%$ of queries. At $B_{\text{expert}} = 50$ KB per expert, the initial download is ≤ 50 MB—feasible for any connected device.*

Proof. Under Zipf(s), the top-M nodes capture probability mass $\sum_{k=1}^{M} k^{-s} / \sum_{k=1}^{\infty} k^{-s} = H_M(s)/\zeta(s)$ where $H_M(s)$ is the generalized harmonic number. For $s > 1$: $1 - H_M(s)/\zeta(s) = \zeta(s)^{-1} \sum_{k>M} k^{-s} \leq \zeta(s)^{-1} \int_M^{\infty} x^{-s}\, dx = \zeta(s)^{-1} M^{1-s}/(s-1) = O(M^{1-s})$. For $s = 1.5$, $M = 1000$: $O(1000^{-0.5}) \approx 0.03$. □

14 Safebots.ai and the Safebox Ecosystem

The LAWS architecture describes a general principle: certified expert libraries, growing from actual workloads, enabling cheap inference on edge devices with cloud-backed knowledge. The Safebox/Safebots.ai ecosystem [19, 16] is one realization of this principle, adding hardware-attested execution and declarative orchestration to the LAWS framework.

14.1 Architecture Overview

Safebox is a hardware-attested compute platform designed for executing AI workloads with cryptographic accountability. It consists of four node types: (1) *Safe.Cloud* browser clients initiating inference; (2) *Safe.Jets* Node.js routers coordinating workloads; (3) *Safe.Drops* browser-side IndexedDB storage for edge caching; and (4) a smart contract (`OpenClaiming.sol`) on a public blockchain providing payment and attestation anchoring.

Each Safebox instance is a cloud instance booted from a deterministically built AMI (Amazon Machine Image), with all build steps auditable and all remote access vectors removed from the finalized image. Nitro attestation and TPM PCR measurements provide cryptographic proof of the software stack running on each instance. M-of-N auditor co-signing is required before mainnet deployment.

In LAWS terms, a Safebox instance is a *certified LAWS node*: it runs the base model (full inference path) and maintains a local expert library. Cache hits return expert evaluations; cache misses run full inference and potentially create new experts. All outputs are cryptographically signed by the attested hardware, providing an accountability chain from input query to certified output.

14.2 Tools, Policies, and Capabilities

The Safebots.ai orchestration layer exposes LAWS components as first-class declarative entities:

- **Tools** correspond to LAWS expert functions f—cheap computations that can be invoked with extracted parameters $\phi(x)$. Each tool is associated with a trie node (its domain) and a validity radius (its certified applicability).
- **Capabilities** correspond to LAWS expert classes—families of tools sharing structural patterns (linear transforms, lookup tables, arithmetic kernels).
- **Policies** govern when to use a cached tool versus invoke the base model—formally encoding the abort-and-replan threshold τ^* (Theorem 9).

This declarative structure allows workloads to be expressed as trees of capability-constrained steps, executed on attested hardware, with every cache hit and miss cryptographically logged. Unlike general-purpose LLM inference APIs where the computation is opaque, Safebox provides an auditable trail of which experts were used, what their certified validity radii were, and where full model inference was invoked.

14.3 LAWS as the Inference Substrate for Distributed AI

The combination of LAWS and Safebox addresses a practical gap in current AI deployment: the absence of a certified, auditable, bandwidth-efficient pathway for distributing AI inference to edge devices while accumulating workload knowledge centrally.

Current paradigms offer a binary choice: (1) cloud inference (high quality, high latency, high cost, privacy concerns) or (2) on-device models (low quality, zero latency, fixed intelligence). LAWS + Safebox offers a third path: on-device expert evaluation for common cases (low latency, low cost, high privacy), full cloud inference for novel cases (high quality when genuinely needed), and continuous improvement of the edge experts from fleet-wide workload experience.

The Safebox platform is the subject of ongoing independent research and development; we refer the reader to [19, 16] for technical details. We note here only that the LAWS theoretical framework developed in this paper provides the formal foundation for the caching, certification, and knowledge-distribution components of that system.

15 Conclusion

We introduced LAWS (Learning from Actual Workloads Symbolically), an inference-time architecture that automatically discovers, certifies, and accumulates reusable computational patterns from deployment experience. Like scientists discovering natural laws from observation, LAWS extracts the invariant regularities in a trained model's behavior—the cheap certified patterns that hold across families of similar inputs—and encodes them as parametrized experts that can be evaluated without running the full model.

LAWS is:

- **Self-certifying:** validity of every expert is proven from model weights alone, with no empirical warmup (Theorem 3).
- **Self-growing:** the expert library grows monotonically (Theorem 6) at sublinear rate $O(2^H \log N)$ (Theorem 7).

- **Energy-efficient:** up to 10× energy reduction for repetitive workloads at 90% hit rate (Theorem 18).
- **Fleet-scalable:** K cooperating units converge $\Omega(K/\log K)$ faster than a single unit (Theorem 15), with OTA updates of $\approx 870\,\mathrm{KB/day}$ per robot for 1,000-unit fleets.
- **Architecture-agnostic:** strictly generalizes KV caching, MoE, and manual symbolic AI (Theorem 10).
- **Biologically grounded:** formalizes Kahneman's System 1/2, Chomsky's innate prior, and expert motor chunking in a single mathematical framework (Section 12).

The key open empirical question is the gap between the worst-case Lipschitz bound $\Lambda(W)$ and the effective in-distribution Lipschitz constant—a gap we expect to be large (10–100×) based on mechanistic interpretability evidence, but which requires systematic measurement. Closing this gap empirically would substantially widen validity radii and improve practical hit rates.

The road from System 2 to System 1 is paved with experience. Scientists have always known this. LAWS makes it formal, provable, and automatically navigated—at the scale of billions of inference queries per day, across every domain where trained neural networks are deployed.

References

[1] Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[2] Peter Belcak and Roger Wattenhofer. Fast feedforward networks. *arXiv preprint arXiv:2308.14711*, 2023.

[3] Kevin Black et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[4] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[5] Anthony Brohan et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[6] Cheng Chi et al. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

[7] Noam Chomsky. *Aspects of the Theory of Syntax*. MIT Press, 1965.

[8] DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference (TCC)*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284, 2006.

[10] Albert Q. Jiang et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[11] Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

[12] Donald E. Knuth. Optimum binary search trees. *Acta Informatica*, 1(1):14–25, 1971.

[13] Woosuk Kwon et al. Efficient memory management for large language model serving with PagedAttention. In *SOSP*, 2023.

[14] Douglas B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[15] Gregory Magarshak. Probabilistic language tries: A unified framework for compression, decision policies, and execution reuse. *arXiv preprint arXiv:2604.06228*, 2026. arXiv:2604.06228 [cs.LG]. Submitted 29 March 2026.

[16] Gregory Magarshak. SafeBox protocol specification: Decentralized encrypted storage with hardware attestation. Technical report, Qbix / Intercoin Research, 2026. Internal specification document.

[17] Gregory Magarshak. Sequential KV cache compression via probabilistic language tries: Beyond the per-vector Shannon limit. *arXiv preprint*, 2026. Companion paper; submitted to arXiv cs.LG concurrently with this work.

[18] Reiner Pope et al. Efficiently scaling transformer inference. In *MLSys*, 2023.

[19] Safebots.ai. Safebots.ai: Hardware-attested AI orchestration. https://safebots.ai, 2026. Platform documentation; retrieved April 2026.

[20] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.

[21] Zhiyuan Wan et al. Symbolic Mixture-of-Experts: Adaptive skill-based routing for heterogeneous reasoning. *arXiv preprint arXiv:2503.05641*, 2025.

[22] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

[23] Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

[24] George Kingsley Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.