Verifiable Explanations Cannot Be Much Smaller Than the Behavior They Explain

Anonymous Authors

Abstract

Interpretability often promises a small explanation of a large model. We study the harder case where the explanation must *stand alone*: it must take inputs, produce outputs, and be verifiable without access to the original model. In that setting, the relevant object is not just a rule but a full executable package, including any decoders, input/output adapters, region selectors, residual corrections, and certificate needed to apply and check it. We formalize the behavior to be explained as finite input–output traces and prove that any exact, verifiable explanation package can be compiled into a simulator for those traces. As a result, it cannot be much smaller, up to constant coding overheads, than the shortest program with the same boundary behavior. This identifies when apparent interpretability compression is real and when it is only hidden in omitted infrastructure: local and approximate explanations help only when the selector, disagreement set, or adapter is itself simple. Experiments on a variety of models show large accounting gaps when these omitted costs are restored. The paper offers a practical takeaway for interpretability evaluation: report the whole ledger, not just the visible artifact.

1 Introduction

Interpretability is often sold as compression: replace a large model with a smaller, more intelligible object. But the visible artifact is often only part of what makes an explanation usable. One still needs input and output conventions, any selector that chooses among local rules, any correction mechanism, and a way to verify the claim. If those pieces are omitted, an explanation can look small only because complexity has been moved off the page.

This paper asks whether an explanation that is *exact*, *executable*, and *independently verifiable* can be substantially smaller than the behavior it explains. In the finite-horizon setting we study, the answer is essentially no. Once all required components are charged—decoders, adapters, selectors, local programs, correction tables, and a certificate—any exact explanation package compiles into a simulator for the same behavior. Theorem 4.1 gives the compilation result, and Theorem 4.2 converts it into a description-length lower bound against boundary-canonical complexity. So an exact standalone explanation cannot be much shorter, up to constant coding overheads, than the shortest program reproducing the same input–output traces.

Two choices make the result meaningful. First, the object being explained is not a parameterization, neuron labeling, or internal coordinate system, all of which can vary under symmetry or reparameterization. We instead work with *boundary behavior*: traces on a fixed protocol. Two mechanisms are equivalent when they induce the same traces. Second, we count the whole executable object, not just the visible summary. A local rule without its selector, a feature dictionary without its decoder, or a surrogate without its correction mechanism is not standalone. We call the complete object an *audit-access package*. Formal definitions appear in Section 3 and Appendix A.

A motivating example. To see why this matters, consider parity on 12 binary inputs. A global exact explanation is just a 12-bit mask plus delimiters. Now suppose a local method partitions the $2^{12} = 4096$ inputs into 16 regions and fits a four-feature parity rule in each region. Counting only the local rules suggests a compact explanation: about $16 \times 4 = 64$ bits. But the package is unusable until we can determine which region applies to each input. If that selector is unstructured, its cost is

Table 1: A toy ledger for a local explanation of a parity-like function.

Object counted	Length	What is being charged
Global exact rule	12 bits	one boundary rule
Local rules only	$16 \times 4 = 64$ bits	the rules themselves
Random selector	$4096 \log_2 16 = 16384$ bits	mapping each input to its region
Local rules plus selector	16448 bits	first fully executable package

about $4096 \log_2 16 = 16384$ bits, overwhelming the rules themselves. Table 1 shows the toy ledger. The lesson is not that locality fails, but that it helps only when the selector is simple.

Contributions. This work makes five contributions:

1. We identify the target of explanation as *boundary behavior*—finite input–output traces—invariant to implementation-level reparameterizations.
2. We define the full *audit-access package*, showing that a standalone explanation must include not just the visible rule but also the selectors, adapters, corrections, and certificate needed for execution and verification.
3. We prove a *certified-collapse theorem*: any exact audit-access package compiles into a standalone simulator, implying a lower bound by boundary-canonical complexity and ruling out large exact self-explanation gains for near-minimal implementations.
4. We characterize the only genuine compression routes—locality, approximation, and quotient structure—and show they help only when the selector, disagreement set, or hidden-to-boundary adapter is simple; matching bounds make tradeoffs tight up to logarithmic terms.
5. We validate the accounting across Boolean functions, automata, MLPs, CNNs, and language models, where compression largely disappears once omitted ledger costs are restored.

2 Related work

Local and additive explanations. LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017) formalize widely used local explanatory objects. Our point is orthogonal to their fitting objectives: a local object becomes an executable explanation only when its validity domain and selection rule are specified. Integrated Gradients (Sundararajan et al., 2017), LRP (Bach et al., 2015), Grad-CAM (Selvaraju et al., 2017), and TCAV (Kim et al., 2018) provide attribution or concept-level summaries. These can be useful diagnostics, but exact certification of boundary behavior also requires adapters and verification.

Explanation correctness, robustness, and failure modes. Prior work argues that interpretability needs formal correctness notions (Doshi-Velez and Kim, 2017; Haufe et al., 2024; Lipton, 2018; Rudin, 2019). Lipton (2018) characterises "interpretability" as task-dependent; we sharpen this for executable certification with an exact description-length lower bound. Rudin (2019) argues for inherently interpretable models over post-hoc explanations of opaque ones; in our ledger, that is the regime with zero adapter and zero correction cost. Sanity checks for saliency maps (Adebayo et al., 2018), robustness tests (Alvarez-Melis and Jaakkola, 2018; Crabbé and van der Schaar, 2023), ROAR-style faithfulness measures (Hooker et al., 2019), infidelity and sensitivity metrics (Yeh et al., 2019), and adversarial attacks on explanation systems (Slack et al., 2020) show that plausible explanations can fail under perturbation or manipulation. The ledger is complementary: those metrics test whether a visible explanation tracks model behaviour, while the ledger tests whether the full package needed to deploy and certify it is compact.

Mechanistic interpretability and internal structure. Mechanistic interpretability studies circuits, causal interventions, superposition, sparse features, and representation structure (Elhage et al., 2022; Bricken et al., 2023; Templeton et al., 2024; Wang et al., 2023; Nanda et al., 2023; Meng et al., 2022). Causal abstraction relates high-level variables to low-level mechanisms (Beckers and Halpern, 2019; Beckers et al., 2020; Geiger et al., 2025; Li et al., 2025). Our framing is compatible with these efforts but distinguishes internal discovery from executable boundary certification. A circuit or feature dictionary can be part of an explanation package, but the decoder, basis convention, boundary adapter, route, and certificate must still be charged.

Gauge, symmetry, and representation dependence. Deep networks admit permutation symmetries, rebasining, and broader representation transformations (Godfrey et al., 2022; Ainsworth et al., 2023).

Coordinate-level explanation scores can therefore change while boundary behavior stays fixed. Rather than making every internal score invariant, we define the target object as the boundary-equivalence class. Internal explanations then enter through explicit adapters and certificates.

MDL, Kolmogorov complexity, and compression. The accounting ledger is inspired by Kolmogorov complexity and MDL (Kolmogorov, 1965; Solomonoff, 1964; Chaitin, 1966; Rissanen, 1978; Grünwald, 2007; Grünwald and Roos, 2019; Barron et al., 1998; Li and Vitányi, 2008). The standalone-cost decomposition of §3 is a two-part MDL code in the sense of Grünwald (2007); Barron et al. (1998): the visible program is the model code, while selectors, adapters, corrections, and certificates are the data code. Blier and Ollivier (2018) apply two-part MDL to deep networks and find that uncompressed weight counts overstate description length; our orthogonal point is that omitted infrastructure understates explanation length. We specialize the usual shortest-description intuition to verifiable explanations of model behavior and identify the missing ledger terms: selectors, adapters, corrections, and certificates.

Computational hardness of explanation. Even when a short explanation *exists*, finding it can be intractable. Wäldchen et al. (2021) show that finding a minimum sufficient reason for a binary classifier decision is Σ_2^p-hard. Our framework is silent on search and focuses on description length; the two are complementary. An auditor with bounded compute also faces the hardness identified by Wäldchen et al. (2021).

Mealy machines and finite-state minimization. The boundary mechanism of Definition 3.5 is a finite-horizon Mealy machine (Mealy, 1955); the boundary equivalence relation is the Nerode-style equivalence on $\mathcal{Q}_n$. Hopcroft minimization (Hopcroft, 1971) and bisimulation (Park, 1981; Milner, 1989) provide the constructive certificate algorithms used in the automata experiments. The full-cost ledger generalizes this finite-state machinery to arbitrary executable explanation packages.

Distillation, compression, and automata certificates. Knowledge distillation and model compression (Buciluǎ et al., 2006; Hinton et al., 2015; Cho and Hariharan, 2019) can produce smaller predictors. In our language, successful distillation finds a shorter boundary program and is therefore a benign exception when it is exact or when approximation is explicit. Finite-state minimization, bisimulation, and model checking (Hopcroft, 1971; Park, 1981; Milner, 1989; Clarke et al., 1999) provide concrete certificate mechanisms. These settings show that full-ledger accounting is not only negative: when quotient structure exists, compact certified explanations are possible and measurable.

3 Boundary mechanisms and explanation ledgers

We separate what a mechanism *does* at its boundary from how it is implemented. For our purposes, internal parameters, neuron bases, and state representations matter only insofar as they affect input–output traces on the audited protocol. The formal definitions below make this precise: implementations are equivalent when they induce the same traces, and the cost of an explanation is the length of the package needed to reproduce them.

Definition 3.1 (Boundary mechanism)**.** A finite-horizon boundary mechanism (equivalently a finite-horizon Mealy machine (Mealy, 1955)) is a tuple $M = (\mathcal{I}, \mathcal{A}, \mathcal{S}, s_0, T, O)$ with input alphabet $\mathcal{I}$, output alphabet $\mathcal{A}$, state space $\mathcal{S}$, transition map $T : \mathcal{S} \times \mathcal{I} \to \mathcal{S}$, and readout $O : \mathcal{S} \to \mathcal{A}$. For a protocol $q = (i_1, \ldots, i_\ell)$ with $\ell \leq n$, $\mathrm{Tr}_n(M, q)$ denotes the emitted output trace. The protocol family $\mathcal{Q}_n \subseteq \cup_{\ell \leq n} \mathcal{I}^\ell$ specifies the finite-horizon behaviors under audit.

Informally, the auditor only cares about the finite set of boundary traces induced on $\mathcal{Q}_n$, whether that protocol family is exhaustive or sampled from a distribution.

Definition 3.2 (Finite-horizon boundary equivalence)**.** Two mechanisms are boundary-equivalent, written $M \equiv_{B,n} M'$, if $\mathrm{Tr}_n(M, q) = \mathrm{Tr}_n(M', q)$ for every $q \in \mathcal{Q}_n$. The equivalence class is denoted $[M]_{B,n}$.

That is, two implementations are indistinguishable to an auditor if they produce the same outputs for every input in the protocol family. We collapse such implementations into the same equivalence class $[M]_{B,n}$ because, from the auditor's perspective, they have the same boundary behaviour.

Definition 3.3 (Boundary-canonical complexity)**.** Fix a universal prefix machine U and a time budget

$t(n)$. The boundary-canonical complexity of $[M]_{B,n}$ is

$$C_n^{B,t}([M]_{B,n}) = \min_P\{|P| : U(P,q) = \mathrm{Tr}_n(M,q)\ \forall q \in \mathcal{Q}_n,\ \mathrm{time}(P,q) \le t(n)\}.$$

In other words, $C_n^{B,t}([M]_{B,n})$ is the length of the shortest time-bounded program that reproduces the audited traces of M. The time bound rules out degenerate encodings that hide cost in unbounded search, and changing the universal machine alters $C_n^{B,t}$ only by an additive constant.

Proposition 3.4 (Boundary invariance)**.** *If $M \equiv_{B,n} M'$, then $C_n^{B,t}([M]_{B,n}) = C_n^{B,t}([M']_{B,n})$.*

Definition 3.5 (Audit-access explanation package)**.** An audit-access explanation package for M on $\mathcal{Q}_n$ is a tuple

$$\Pi = (E, D, \mathrm{Enc}_{\mathrm{in}}, \mathrm{Dec}_{\mathrm{out}}, G, \mathit{Sel}, \mathit{Corr}, V, \sigma),$$

where E is an emitted transcript, D is a decoder, $\mathrm{Enc}_{\mathrm{in}}$ and $\mathrm{Dec}_{\mathrm{out}}$ are boundary adapters, G is one or more executable explanation programs, Sel is a selector for local packages, Corr is a correction object for approximate packages, V is a certificate, and σ is a self-delimiting schema describing the ledger. Components not used by a particular package may be null but must be explicitly represented.

Intuitively, an audit-access package contains everything needed to run and verify the explanation without the original model; Appendix A.3 gives the full component breakdown.

Definition 3.6 (Standalone cost)**.** The *standalone cost* of an audit-access package Π is the total length (in bits) of everything an auditor must read in order to run and check the explanation:

$$\begin{aligned} K_{\mathrm{stand}}(\Pi) = {} & |\sigma| + |E| + |D| + |\mathrm{Enc}_{\mathrm{in}}| + |\mathrm{Dec}_{\mathrm{out}}| + |G| \\ & + |\mathit{Sel}| + |\mathit{Corr}| + |V| + O(\log n). \end{aligned}$$

The $O(\log n)$ term accounts for self-delimiting headers and the encoding of the horizon n. When a time budget matters, we write K_{stand}^t and require each executable component of Π to finish within $t(n)$ steps. Intuitively, $K_{\mathrm{stand}}(\Pi)$ is the number of bits needed to write down a complete explanation package so that a third party can run it without access to the original mechanism.

Exactness, locality and approximation. A package is *exact* if it matches M on every audited input, *local* if it uses region-specific rules with a selector, and *ϵ-approximate* (under μ_n) if it errs on at most an ϵ fraction of inputs. Approximate packages become exact by adding a correction object; otherwise the claim is only distributional. Appendix A discusses scope and non-executable explanations.

4 Exact certified explanations collapse to simulators

We now formalise the core negative result. Any exact, executable, independently verifiable explanation package can be turned into a simulator by running its components in sequence, so it cannot be much smaller than the behaviour it certifies. The following theorems make this precise.

Theorem 4.1 (Certified explanation collapse)**.** *Let Π be an exact audit-access explanation package for M on $\mathcal{Q}_n$ whose executable components halt within the public time budget $t(n)$. Suppose D, together with the public schema σ, decodes the emitted transcript E into executable components and V verifies equality of the decoded output with $\mathrm{Tr}_n(M,q)$ for all $q \in \mathcal{Q}_n$. Then there exists a standalone simulator S_Π such that*

$$S_\Pi(q) = \mathrm{Tr}_n(M,q) \quad \forall q \in \mathcal{Q}_n,$$

and

$$|S_\Pi| \le K_{\mathrm{stand}}^t(\Pi) + O(\log n).$$

The package already contains everything needed to reproduce the audited outputs: schema, decoder, adapters, program, selector, correction, and certificate. Executing those components yields a simulator, so an exact explanation is simply a simulator written in ledger form.

Theorem 4.2 (Boundary-canonical lower bound)**.** *For every exact certified explanation package* Π *of* M *on* $\mathcal{Q}_n$ *whose executable components halt within* $t(n)$*,*

$$K^t_{\mathrm{stand}}(\Pi) \geq C^{B,t}_n([M]_{B,n}) - O(\log n).$$

This is the full-cost accounting lower bound: no exact, verifiable explanation can be much shorter than the shortest program with the same boundary behavior. If a package appears far smaller than $C^{B,t}_n([M]_{B,n})$, then either the implementation was redundant or the package has relaxed or omitted something—for example exactness, executability, certification, or some required adapter, selector, correction, or other ledger component.

Definition 4.3 (η-minimal implementation)**.** A mechanism M is η-minimal at horizon n if its implementation complexity satisfies

$$K_{\mathrm{impl}}(M) \leq C^{B,t}_n([M]_{B,n}) + \eta.$$

Here $K_{\mathrm{impl}}(M)$ is the prefix length of a runnable implementation of M under the same boundary convention and time budget.

Theorem 4.4 (No substantially shorter exact self-explanation)**.** *If* M *is* η*-minimal at horizon* n*, then every exact self-emitted certified explanation package* Π *whose executable components halt within* $t(n)$ *satisfies*

$$K^t_{\mathrm{stand}}(\Pi) \geq K_{\mathrm{impl}}(M) - \eta - O(\log n).$$

Thus a near-minimal mechanism cannot emit an exact, certified self-explanation that is much shorter than itself. The result is only about exact, global, executable certificates: approximate summaries, partial circuits, lossy probes, feature visualizations, and other human-facing narratives may still be useful, but once upgraded to exact certification of the full finite-horizon behavior, they inherit simulator-level description length.

Appendix B records the redundancy exception: shorter explanations can reflect a simpler boundary program, not a violation of the lower bound.

Where certificates enter. Our proofs assume only that the certificate is strong enough to verify equality of decoded outputs with the ground truth on $\mathcal{Q}_n$. For deterministic finite-state systems this can be done exactly with bisimulation or quotient certificates. For learned systems one typically relies on empirical audit artifacts—saved traces, held-out evaluations, and hash commitments for the audited slice—rather than a proof of global equivalence. In those cases the reported ledger certifies exactness only on the audited finite protocol family, and should be read as a measurement proxy for the theoretical quantity rather than a mathematical proof of behavior outside that audit set.

Proposition 4.5 (Finite-state certificate classes)**.** *For deterministic finite-horizon finite-state systems, bisimulation certificates and quotient certificates instantiate audit access. Their ledger consists of the relation or quotient map, representative transitions, boundary adapters, and verifier convention.*

5 Locality and approximation: when do they help?

Many widely used interpretability methods are not global simulators. They may work only on a small region of input space (*local* explanations), agree with the model only on average (*approximate* explanations), or operate on hidden coordinates rather than boundary inputs. Our framework does not dismiss these approaches. Instead, it tells us when they yield real compression: only when the extra infrastructure they require is itself succinct. This section makes that precise.

5.1 Local explanations must pay for selection

Let $\mathcal{X}_n$ denote the finite set of audited boundary inputs. A local package consists of regions $\mathcal{R}_1, \ldots, \mathcal{R}_k$, local programs $P_1, \ldots, P_k$, and a selector $Sel : \mathcal{X}_n \to [k]$. Exact execution requires $P_{Sel(q)}(q) = \mathrm{Tr}_n(M, q)$ for every audited q.

Theorem 5.1 (Local accounting bound)**.** *For any exact local certified package* Π_{loc}*,*

$$K_{\mathrm{stand}}(\Pi_{\mathrm{loc}}) \geq K(Sel) + \sum_{j=1}^{k} K(P_j) + K(\mathrm{Enc}_{\mathrm{in}}, \mathrm{Dec}_{\mathrm{out}}) + K(V) - O(\log n).$$

Consequently, a local explanation can only beat a global boundary program if the cost of specifying the selector, adapters, certificate and local programs together is smaller than $C_n^{B,t}([M]_{B,n})$.

The critical term is $K(Sel)$. Locality helps only when the selector has a short boundary description; arbitrary hidden clusters or post-hoc neighbourhoods can make selector cost dominate and erase any savings.

Proposition 5.2 (Random selectors are incompressible)**.** *Let $\mathcal{X}_n$ have size N and choose $Sel : \mathcal{X}_n \to [k]$ uniformly at random. With probability at least $1 - 2^{-c}$,*

$$K(Sel) \geq N \log_2 k - c - O(1).$$

Random selectors are essentially incompressible. A partition that looks simple in hidden coordinates may still lack a short boundary description, so without a cheap boundary selector, locality is not an operational compression.

5.2 Approximation must pay for disagreement or weaken the target

Approximate explanations can be excellent scientific tools, but an approximate package is not an exact explanation until disagreements are either accepted as part of a distributional claim or repaired. Let A be an approximate executable package and define the disagreement set

$$D_A = \{q \in \mathcal{Q}_n : A(q) \neq \mathrm{Tr}_n(M, q)\}.$$

A correction object may encode D_A and corrected outputs, or it may encode a shorter program that repairs the disagreements.

Theorem 5.3 (Approximation–correction tradeoff)**.** *For any approximate package A and any correction object $Corr$ that makes it exact on $\mathcal{Q}_n$,*

$$K_{\mathrm{stand}}(A, Corr) \geq K(A) + K(D_A \mid A) + K(\mathrm{Tr}_n(M, \cdot)|_{D_A} \mid A, D_A) - O(\log n).$$

If A has error rate ϵ under the uniform distribution on a finite domain of size N, the boundary output alphabet has size $|\mathcal{A}|$, and the disagreement locations are otherwise unstructured, then the correction locations and corrected labels together cost at least

$$\log_2 \binom{N}{\epsilon N} + \epsilon N \log_2(|\mathcal{A}| - 1) = N H_2(\epsilon) + \epsilon N \log_2(|\mathcal{A}| - 1) + O(\log N)$$

bits.

Thus approximation helps only when the disagreement set is small, structured, or accepted as part of a distributional claim. Otherwise, exactness requires a correction object, and if disagreements are unstructured, the locations already cost about $N H_2(\epsilon)$ bits, with an additional label term for non-binary boundary alphabets.

5.3 Internal-coordinate explanations require boundary adapters

Many mechanistic explanations live in hidden coordinates. To become executable boundary explanations, they must include input and output adapters linking internal structure to boundary traces.

Proposition 5.4 (Coordinate scores are not boundary invariants)**.** *Any score defined on raw hidden coordinates can change under an invertible reparameterization that preserves boundary traces, unless the score is explicitly quotient-invariant or paired with a boundary certificate.*

The proposition shows raw hidden-coordinate scores are not boundary properties unless they are quotient-invariant or certified. Mechanistic insight can still be valuable, but certified explanations must also pay for the adapters and verification linking internal structure to boundary behavior.

6 Empirical audit

Sections 4 and 5 give worst-case exact results. Here we test the same accounting effects in practice using a common ledger for visible, selector, adapter, correction, certificate, and total cost. We compare visible-only and full-cost rankings. When rankings differ, compression has been outsourced to hidden infrastructure rather than earned. For the learned-model audits, the certificate term is an empirical audit artifact for the fixed finite protocol family we save—not a proof of global behavioral equivalence outside that audited set.

Experiment 1: Boolean teachers. We use simple Boolean teachers to isolate selector and correction costs: sparse parity, DNF, sparse lookup tables, and uniform random functions on $d \in \{8, 10, 12, 14\}$ bits. We compare five explanation families: a global rule, local rules with structured, random, or hash selectors, and approximate local rules with explicit correction. Table 2 shows the d=12 slice. By visible cost alone, random- and hash-selected local packages look cheap. Under full accounting, selector and correction costs raise them by over an order of magnitude, leaving the global package first and the learned structured selector second, as predicted by Theorem 5.1.

Table 2: Experiment 1 at d=12, averaged across seeds and teachers. Package names are literal families; k=8 means eight regions. "Vis. fid." is fidelity before correction; after correction, every package is exact. Visible-only cost ranks local hash first, but full cost ranks global exact first (640 bits), then local learned, with random and approximate-local packages far worse. The approximate row intentionally matches the random-selector baseline on this slice because it uses the same visible rule and pays its disagreements back via an explicit correction object.

Package	visible	selector	adapter	correction	certificate	full	vis. fid.
Global exact rule	632	0	0	0	8	640	1.000
Local rule + structured selector (k=8)	32	344	0	4564	8	4948	0.671
Local rule + random selector (k=8)	32	12288	0	4142	8	16470	0.698
Local rule + hash selector (k=8)	32	64	0	4686	8	4790	0.655
Local rule + learned selector (k=8)	32	560	0	4054	8	4654	0.710
Approx. local rule + correction	32	12288	0	4142	8	16470	0.698

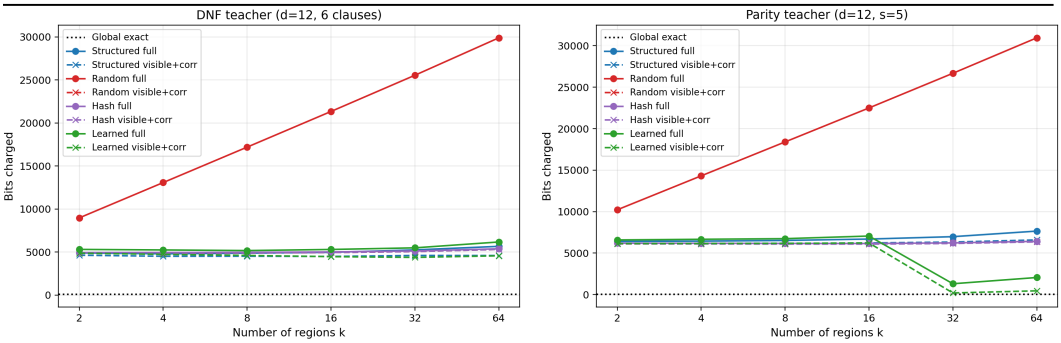

Figure 1: Selector sweep for parity and DNF teachers. The key comparison is dashed visible-plus-correction curves versus solid full-cost curves: as k grows, local rules get simpler, but only structured and learned selectors keep total cost close to the visible line. Random selectors rise sharply because routing cost dominates, so omitted selector bits can reverse explanation rankings.

Experiment 2: selector sweep. To isolate selector cost, we fix two Boolean teachers—parity and a six-clause DNF on d=12—and vary the number of regions $k \in \{2, 4, 8, 16, 32, 64\}$. Figure 1 compares structured, random, hash, and learned selectors, plotting visible-plus-correction cost (dashed) against full cost (solid). As k grows, visible cost falls because local rules simplify, but only structured and learned selectors keep total cost close. Random selectors rise at the $N \log_2 k$ rate from Proposition 5.2, while hash selectors still pay full correction cost. The learned selector performs best at large k because it compresses routing more efficiently than block partitioning.

Experiment 3: neural teachers. We audit small MLP teachers on parity, sparse conjunctions, two moons, and a random-label control. We compare a global tree surrogate, a small distillation network, local linear probes in hidden space, and the same probes routed from the input side by a shallow tree. Figure 2 shows the ledger. Hidden-local packages have almost no selector cost but pay heavily in adapter bits because cluster assignments must be recorded for each input. The boundary-routed variant shifts those bits into a visible selector and, except on the random control, matches or improves on full cost. No free hidden local explanations: one pays for cluster assignments or a boundary router.

Experiment 4: finite-state machines. We audit deterministic finite-state transducers with input alphabets $\{2, 4\}$, output alphabets $\{2, 8\}$, and state counts $|Q| \in \{32, 64, 128, 256, 512, 1024, 2048\}$. We compare random automata with planted-structure automata whose states share transitions in quotient classes. The packages are a raw transition table, quotient certificate, and trace simulator. Figure 3 shows quotient certificates beat raw tables on planted automata, with the gap growing in $|Q|$. On random automata there is little structure to exploit, so the quotient certificate gets more expensive once the quotient map is charged. The trace simulator stays cheap as it certifies truncated behavior.

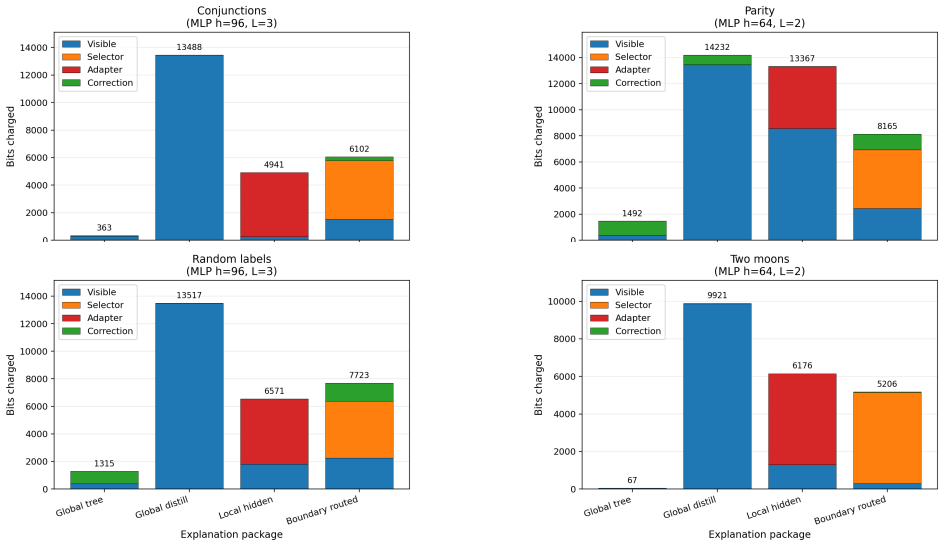

Figure 2: Full-cost ledger for feed-forward neural teachers across four panels. Certificate cost is nonzero but too small to see, so it is omitted from the bars and counted only in the total shown on top.

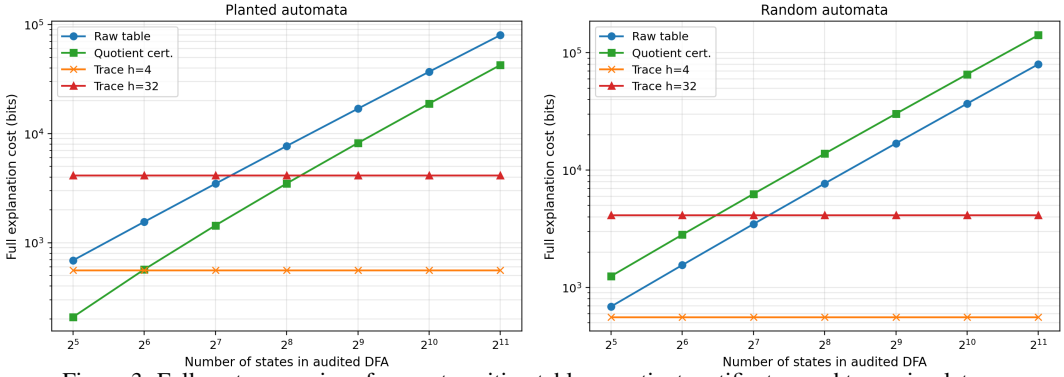

Figure 3: Full-cost comparison for raw transition tables, quotient certificates, and trace simulators.

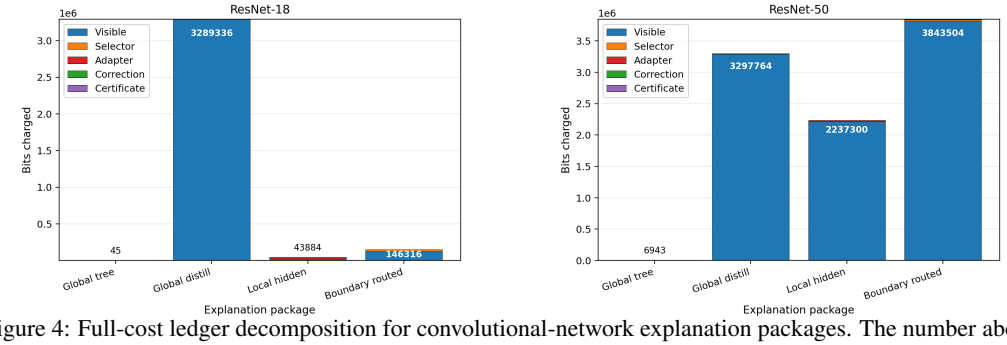

Figure 4: Full-cost ledger decomposition for convolutional-network explanation packages. The number above each bar is total cost. Certificate cost is included in the stacked bars, though visually tiny relative to the dominant visible, selector, and adapter terms. For ResNet-18, the near-zero correction cost of the global tree occurs because the audited teacher slice is nearly constant; it should not be read as a generic statement about exact global trees for natural CNN decision boundaries.

Experiment 5: convolutional networks. We audit ResNet-18 and ResNet-50 on a synthetic CIFAR-style task, reusing the four explanation families from Experiment 3. Figure 4 decomposes full cost. On ResNet-18 the hidden-state package saves probe bits but pays about 22,800 bits for cluster assignments; the boundary-routed package pays a similar amount in selector bits. On ResNet-50 the adapter cost stays similar because the codebook compresses cluster IDs rather than hidden vectors, while visible probe cost grows with hidden dimension. The same hidden-versus-visible tradeoff persists: charging the router shifts bits rather than eliminating them. The unusually tiny ResNet-18 global-tree bar reflects a degenerate audited slice on which the trained teacher's boundary trace

collapses to a nearly constant label; we therefore treat it as a redundancy case rather than evidence that generic CNN boundaries admit 45-bit exact trees.

Experiment 6: language models. We audit a ladder of pretrained decoder-only transformers on a multi-choice lexical-similarity task. Figure 5 reports the full-cost ledger for each package and model plus the scaling trend across the ladder, while Appendix Figure 6 isolates smaller charged terms compressed in the main plot. Three patterns stand out. First, hidden-package adapter cost stays nearly constant at about 2–3 kilobits because only the cluster-ID codebook is transmitted. Second, visible probe cost for hidden-local packages is roughly linear in hidden dimension once we move past the two smallest 768-dimensional outliers, rising from 262,656 bits at 1024 dimensions to 917,984 bits at 4096 dimensions. Boundary-routed local cost, by contrast, stays near 30,000 bits. Third, the global tree surrogate remains cheap, while the neural distillation surrogate is largely fixed by its architecture rather than teacher size. Across the ladder, total cost grows far more slowly than raw model size and is driven mainly by package choice and audit interface, matching Theorem 4.2.

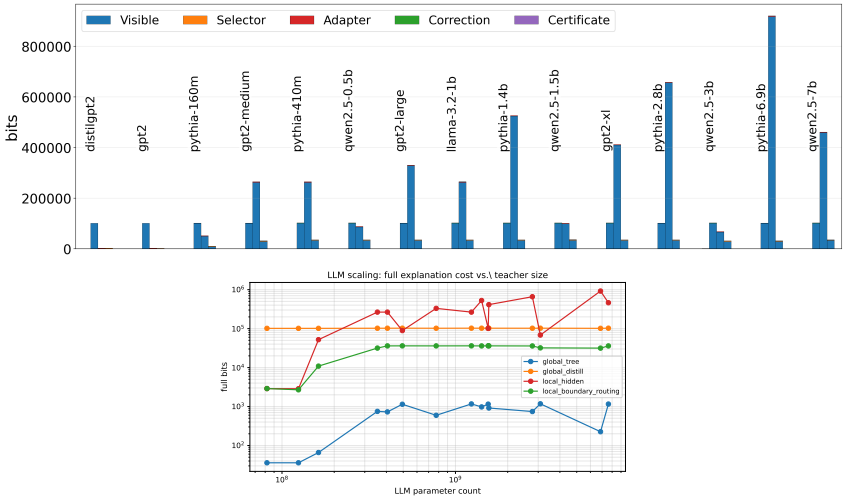

Figure 5: LLM explanation costs. Top: cost ledger by package and model. Bottom: explanation cost versus teacher size. For a full breakdown of the charged components, see Appendix Figure 6.

7 Limitations

The theory is finite-horizon and exact, so broader stochastic, interactive, infinite-horizon, or adversarial settings require additional assumptions and are not covered by the main theorem as stated. The empirical audits also rely on MDL-style code-length proxies. For neural and language-model experiments, certificate cost is usually only an empirical audit artifact on a finite protocol rather than a mathematical proof of global behavioral equivalence. More fundamentally, the framework evaluates standalone executable certification, not whether an explanation is cognitively useful, scientifically insightful, or causally revealing to a human. Finally, we give lower bounds on package size, not algorithms for finding optimal packages; in the worst case, the search problem can still be intractable.

8 Conclusion

Full-ledger accounting turns self-explanation into a precise boundary theorem. Exact, executable, independently verifiable explanations of finite-horizon behavior compile into simulators, so their standalone cost is lower bounded by the boundary-canonical complexity of the behavior they certify. The bound is tight up to logarithmic terms along the escape routes we characterize—structured locality, approximation with correction, and genuine quotient structure—and the empirical audits show the predicted rank reversals across Boolean teachers, automata, MLPs, ResNets, and a language-model ladder. The practical message is: if an explanation is meant to stand alone, the relevant object is the whole ledger, not just the visible artifact. We release the accompanying code with the paper. Future work should broaden the framework in both theory and experiment: theoretically, by extending the lower bounds to more distributional, game-theoretic, interactive, and adversarial notions of audit access; experimentally, by instantiating the ledger on broader protocol families and more certificate regimes, including mechanistic circuits and agentic multi-turn tasks.

References

J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. *NeurIPS*, 2018.

S. Ainsworth, J. Hayase, and S. Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *ICLR*, 2023.

D. Alvarez-Melis and T. S. Jaakkola. On the robustness of interpretability methods. *arXiv:1806.08049*, 2018.

S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 2015.

A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 1998.

S. Beckers and J. Y. Halpern. Abstracting causal models. *AAAI*, 2019.

S. Beckers, F. Eberhardt, and J. Y. Halpern. Approximate causal abstraction. *UAI*, 2020.

L. Blier and Y. Ollivier. The description length of deep learning models. *NeurIPS*, 2018.

T. Bricken, A. Templeton, J. Batson, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. *KDD*, 2006.

G. J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 1966.

J. H. Cho and B. Hariharan. On the efficacy of knowledge distillation. *ICCV*, 2019.

P. Christiano, A. Cotra, and M. Xu. Eliciting latent knowledge: How to tell if your eyes deceive you. *ARC technical report*, 2021.

E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

J. Crabbé and M. van der Schaar. Evaluating the robustness of interpretability methods through explanation invariance and equivariance. *NeurIPS*, 2023.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv:1702.08608*, 2017.

N. Elhage, T. Hume, C. Olsson, et al. Toy models of superposition. *Transformer Circuits Thread*, 2022.

A. Geiger, D. Ibeling, A. Zur, M. Chaudhary, S. Chauhan, J. Huang, A. Arora, Z. Wu, N. Goodman, C. Potts, and T. Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *JMLR*, 2025.

C. Godfrey, D. Brown, T. Emerson, and H. Kvinge. On the symmetries of deep learning models and their internal representations. *NeurIPS*, 2022.

P. D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

P. D. Grünwald and T. Roos. Minimum description length revisited. *International Journal of Mathematics for Industry*, 2019.

G. H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 1955.

S. Wäldchen, J. Macdonald, S. Hauch, and G. Kutyniok. The computational complexity of understanding binary classifier decisions. *Journal of Artificial Intelligence Research*, 2021.

S. Haufe, R. Wilming, B. Clark, R. Zhumagambetov, D. Panknin, and A. Boubekki. Explainable AI needs formal notions of explanation correctness. *arXiv:2409.14590*, 2024.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *NIPS Workshop*, 2015.

S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim. A benchmark for interpretability methods in deep neural networks (ROAR). *NeurIPS*, 2019.

J. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In *Theory of Machines and Computations*, 1971.

B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viégas, and R. Sayres. Interpretability beyond feature attribution: TCAV. *ICML*, 2018.

A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1965.

X. Li, S.-O. Kaba, and S. Ravanbakhsh. On the identifiability of causal abstractions. *AISTATS*, 2025.

M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 2008.

Z. C. Lipton. The mythos of model interpretability. *Queue*, 2018.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *NeurIPS*, 2017.

K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and editing factual associations in GPT. *NeurIPS*, 2022.

R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability. *ICLR*, 2023.

D. Park. Concurrency and automata on infinite sequences. *LNCS*, 1981.

M. T. Ribeiro, S. Singh, and C. Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. *KDD*, 2016.

J. Rissanen. Modeling by shortest data description. *Automatica*, 1978.

C. Rudin. Stop explaining black box machine learning models for high-stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 2019.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM. *ICCV*, 2017.

D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling LIME and SHAP. *AIES*, 2020.

R. J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 1964.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *ICML*, 2017.

A. Templeton, T. Conerly, J. Marcus, et al. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. *Transformer Circuits Thread*, 2024.

K. R. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. *ICLR*, 2023.

C.-K. Yeh, C.-Y. Hsieh, A. S. Suggala, D. I. Inouye, and P. Ravikumar. On the (in)fidelity and sensitivity of explanations. *NeurIPS*, 2019.

A Full formalism

This appendix expands the definitions used in the main text. The purpose is to remove ambiguity about what is counted and what is not counted.

A.1 Universal machine, coding convention, and time budget

Fix a self-delimiting universal prefix machine U. A codeword p is valid if U can parse its schema without external side information. All composite objects use self-delimiting tuples, so concatenation costs an additive $O(\log m)$ term for m components and horizon encodings. For a horizon n, the machine is restricted to time $t(n)$ per audited protocol. The time bound is not central to the compression argument, but it prevents descriptions that hide the boundary behavior inside an unbounded proof search.

A boundary program for M is a code P such that $U(P, q) = \mathrm{Tr}_n(M, q)$ for every $q \in \mathcal{Q}_n$. A package program may contain multiple executable objects: a selector, local programs, adapters, corrections, and verifiers. The standalone cost is the size of the entire package plus the fixed interpreter required to execute the package under the public convention.

A.2 Boundary trace spaces

For protocol $q = (i_1, \ldots, i_\ell)$, define states $s_j = T(s_{j-1}, i_j)$ and outputs $a_j = O(s_j)$. The boundary trace may be the full sequence $(a_1, \ldots, a_\ell)$ or a task-specific projection. If the trace is projected, the projection map is part of the public boundary convention; if it is learned or model-specific, it must be included in the package.

The protocol family $\mathcal{Q}_n$ can be exhaustive, distributional, or adversarially specified. The exact theorems use a finite set $\mathcal{Q}_n$ or a decidable family over which the certificate verifies equality. Empirical audits estimate the corresponding costs using sampled protocols and report the sampling distribution separately.

A.3 Explanation package schema

The schema σ contains a list of component types, coding rules, and null markers. The emitted transcript E is what the model or explanation generator provides. The decoder D maps E into executable objects. The input adapter $\mathrm{Enc}_{\mathrm{in}}$ maps boundary protocols into the domain expected by the explanation. The output adapter $\mathrm{Dec}_{\mathrm{out}}$ maps the explanation output back into the boundary trace convention. The selector Sel maps protocols or adapted inputs to a local index. The correction object $Corr$ repairs approximate outputs. The verifier V checks exact equality or a declared weaker property.

Equivalently, an audit-access package bundles every object an external auditor would need to execute the explanation from scratch:

- a transcript E emitted by the explanation generator and a decoder D that turns that transcript into executable code;
- input and output adapters $\mathrm{Enc}_{\mathrm{in}}$ and $\mathrm{Dec}_{\mathrm{out}}$ that translate between the boundary protocol and whatever domain the explanation operates on;
- one or more visible programs G that actually compute the explanation's predictions; in the local case G is a list of region-specific rules;
- a selector Sel that tells us which local rule to use for a given input;
- a correction object $Corr$ that patches residual disagreements when the package is approximate;
- a certificate V that convinces the auditor that the outputs match the ground truth;
- and a schema σ so that the auditor knows where each component begins and ends.

Components that are unused by a particular package are still represented explicitly through the schema's null markers so that the package remains self-delimiting.

A common undercount is to treat D, $\mathrm{Enc}_{\mathrm{in}}$, $\mathrm{Dec}_{\mathrm{out}}$, or Sel as "obvious." They are obvious only if they are public constants shared across all mechanisms in the comparison class. If they are fitted to the mechanism, they are part of the explanation package.

A.4 Exact, approximate, and distributional claims

An exact claim is universal over $\mathcal{Q}_n$. A distributional claim is parameterized by a distribution μ_n and tolerance ϵ:

$$\mathbb{P}_{q\sim\mu_n}[A(q) \neq \mathrm{Tr}_n(M, q)] \leq \epsilon.$$

A repaired approximate claim adds a correction object and becomes exact.

Our results concern explanations that are independently executable and verifiable. This includes finite-state quotient certificates, executable surrogates, distilled models with saved decoders, local systems with explicit routers, and logged pipelines whose decoded outputs can be checked against boundary traces. Human-facing artifacts that cannot be run on their own, such as a heatmap without a predictive procedure, may still be useful, but they lie outside the certified-collapse theorem because they do not define a simulator.

A.5 Local package normal form

Any local package can be represented in normal form as

$$\Pi_{\mathrm{loc}} = (\sigma, E, D, \mathrm{Enc}_{\mathrm{in}}, \mathrm{Dec}_{\mathrm{out}}, \mathit{Sel}, \{P_j\}_{j=1}^{k}, \mathit{Corr}, V).$$

The selector may be explicit as a table, implicit as a program, or induced by a learned model. Its cost is the shortest code that implements the selector under the same boundary convention. Hidden-state clustering is not free: if a cluster assignment requires running the original model, then either the original model is part of the explanation package or a boundary predictor for the cluster must be supplied.

A.6 Boundary-canonical versus implementation complexity

$C_n^{B,t}([M]_{B,n})$ is a property of the boundary behavior. $K_{\mathrm{impl}}(M)$ is a property of one implementation. If $K_{\mathrm{impl}}(M) \gg C_n^{B,t}([M]_{B,n})$, then the implementation is redundant and a short exact explanation may exist. If $K_{\mathrm{impl}}(M) \approx C_n^{B,t}([M]_{B,n})$, then an exact explanation cannot be much shorter without violating the lower bound.

B Proofs

B.1 Boundary invariance

Proof of Proposition 3.4. If $M \equiv_{B,n} M'$, then the function $q \mapsto \mathrm{Tr}_n(M, q)$ equals the function $q \mapsto \mathrm{Tr}_n(M', q)$ on $\mathcal{Q}_n$. The feasible set in the minimization defining $C_n^{B,t}([M]_{B,n})$ is therefore identical to the feasible set for $C_n^{B,t}([M']_{B,n})$. Equal feasible sets have equal minimum code length. □

B.2 Certified collapse

Lemma B.1 (Package execution lemma)**.** *Given an audit-access package* Π *satisfying Definition 3.5, there exists a universal wrapper* W *of length* $O(1) + O(\log n)$ *that executes the decoded package on any* $q \in \mathcal{Q}_n$ *whenever the decoded components halt within the time budget.*

Proof. The wrapper parses the self-delimiting schema, invokes D on E, parses the decoded components, applies $\mathrm{Enc}_{\mathrm{in}}$ to q, evaluates the global program or $P_{\mathit{Sel}(q)}$, applies Corr if non-null, maps the result through $\mathrm{Dec}_{\mathrm{out}}$, and outputs the boundary trace. The code of the wrapper is fixed by the public convention. The horizon and delimiter overheads are self-delimiting and contribute $O(\log n)$. □

Proof of Theorem 4.1. Let S_Π be the program that hard-codes Π and the wrapper from Lemma B.1. On input q, S_Π executes the package and returns the decoded boundary output. By the certificate assumption, the decoded output equals $\mathrm{Tr}_n(M, q)$ for every $q \in \mathcal{Q}_n$. Therefore S_Π is a standalone simulator of the target boundary behavior. Its length is the length of the hard-coded package plus the wrapper and delimiters, hence at most $K^t_{\mathrm{stand}}(\Pi) + O(\log n)$. □

Proof of Theorem 4.2. By Theorem 4.1, an exact certified package Π yields a standalone simulator S_Π with $|S_\Pi| \leq K^t_{\mathrm{stand}}(\Pi) + O(\log n)$. Since $C_n^{B,t}([M]_{B,n})$ is the minimum length of any standalone simulator of the same boundary behavior that also halts within $t(n)$, $C_n^{B,t}([M]_{B,n}) \leq |S_\Pi|$. Combining the inequalities and rearranging gives $K^t_{\mathrm{stand}}(\Pi) \geq C_n^{B,t}([M]_{B,n}) - O(\log n)$. □

Proof of Theorem 4.4.

$$K_{\mathrm{impl}}(M) \le C_n^{B,t}([M]_{B,n}) + \eta, \qquad C_n^{B,t}([M]_{B,n}) \ge K_{\mathrm{impl}}(M) - \eta.$$

Theorem 4.2 then implies $K^t_{\mathrm{stand}}(\Pi) \ge K_{\mathrm{impl}}(M) - \eta - O(\log n)$. □

Corollary B.2 (Redundancy is the benign exception)**.** *If an implementation M is not near-minimal, a shorter explanation may exist by revealing a simpler boundary program. In that case the explanation compresses implementation redundancy rather than escaping the boundary-canonical lower bound.*

Proof of Corollary B.2. If $K_{\mathrm{impl}}(M) > C_n^{B,t}([M]_{B,n}) + \eta$ for large η, then a shortest boundary program of length $C_n^{B,t}([M]_{B,n})$ may be shorter than the given implementation. An explanation that emits such a program compresses the implementation, not the boundary behavior below its canonical complexity. □

B.3 Local accounting

Lemma B.3 (Selector necessity)**.** *An exact local package over regions $\mathcal{R}_1, \dots, \mathcal{R}_k$ must contain enough information to determine, for every audited protocol q, which local program is executed, unless a single local program is valid for all regions.*

Proof. Suppose two packages have identical local programs and adapters but route some q to different local indices producing different outputs. At most one can be exact for the target trace. Therefore the routing decision is semantically necessary. It can be supplied by a table, program, learned predictor, public convention, or certificate-derived rule, but if it is not public it must be encoded in the package. □

Proof of Theorem 5.1. Put the local package in normal form. By Lemma B.3, the package must encode a selector or an equivalent object. It must also encode each local program that may be executed on the covered domain, the adapters needed to map boundary protocols to local-program inputs and local outputs back to boundary traces, and the certificate. By the chain rule for prefix complexity, the complexity of the whole tuple is at least the sum of the component complexities minus the self-delimiting overhead needed to parse the tuple. Absorbing shared decoding conventions and tuple headers into the $O(\log n)$ term gives the displayed bound. □

Proof of Proposition 5.2. There are k^N selectors from an N-element domain to $[k]$. At most 2^m binary strings have prefix complexity at most m. Therefore the fraction of selectors with $K(Sel) < N \log_2 k - c$ is at most 2^{-c}, up to a universal-machine constant. Equivalently, with probability at least $1 - 2^{-c}$, $K(Sel) \ge N \log_2 k - c - O(1)$. □

B.4 Approximation and correction

Lemma B.4 (Correction decomposition)**.** *Let A be an approximate executable package and D_A its disagreement set on $\mathcal{Q}_n$. Any correction object making A exact must determine both the disagreement locations and the corrected boundary traces on those locations, conditional on A.*

Proof. If a repaired package is exact, then for every $q \notin D_A$ it may return $A(q)$, while for every $q \in D_A$ it must return a different value. Thus the repair mechanism induces a membership test for D_A and a map from each member of D_A to the target trace. If either object is not derivable from A, it must be encoded in the correction object. □

Proof of Theorem 5.3. By Lemma B.4, the repaired exact package contains A, enough information to identify D_A conditional on A, and enough information to output the corrected traces on D_A conditional on A and D_A. Prefix coding gives the first inequality. If the domain has size N, the output alphabet has size $|\mathcal{A}|$, and the unstructured disagreement set has size ϵN, specifying the set requires $\log_2 \binom{N}{\epsilon N}$ bits, while specifying the corrected non-default output symbol at each disagreement costs an additional $\epsilon N \log_2(|\mathcal{A}| - 1)$ bits in the worst case. Stirling's approximation gives the stated entropy term for the locations. □

B.5 Coordinate dependence

Proof of Proposition 5.4. Let a mechanism be represented by two adjacent maps $h = f(x)$ and $y = g(h)$. For any invertible map A, define $h' = Ah$, $f' = Af$, and $g' = gA^{-1}$. The boundary function is unchanged because $g'(f'(x)) = g(A^{-1}Af(x)) = g(f(x))$. However coordinate-level quantities such as sparsity of h, attribution mass assigned to coordinates, or basis-aligned feature counts can change under A. Therefore such scores are not boundary invariants unless the score itself quotients over the transformation class or a boundary certificate fixes the coordinate convention. □

C Certificate constructions

This appendix gives concrete audit-access instantiations. The goal is not to claim that every neural explanation admits these certificates, but to show that the audit assumption is mathematically non-vacuous.

C.1 Bisimulation certificates

For deterministic finite-state systems $M = (\mathcal{I}, \mathcal{A}, \mathcal{S}, s_0, T, O)$ and $M' = (\mathcal{I}, \mathcal{A}, \mathcal{S}', s'_0, T', O')$, a bisimulation certificate is a relation $R \subseteq \mathcal{S} \times \mathcal{S}'$ such that $(s_0, s'_0) \in R$ and for every $(s, s') \in R$ and input $i \in \mathcal{I}$,

$$O(s) = O'(s'), \qquad (T(s,i), T'(s',i)) \in R.$$

The verifier checks these finite conditions. By induction on protocol length, related initial states produce identical traces for all horizons. The certificate cost is the relation encoding plus the transition and output tables required by the verifier if they are not already public.

C.2 Quotient certificates

A quotient certificate for a mechanism M consists of a partition map $\rho : \mathcal{S} \to [m]$, representative transitions $\bar{T} : [m] \times \mathcal{I} \to [m]$, and representative outputs $\bar{O} : [m] \to \mathcal{A}$. The verifier checks that for all states s and inputs i,

$$\bar{O}(\rho(s)) = O(s), \qquad \rho(T(s,i)) = \bar{T}(\rho(s), i).$$

If the checks pass, the quotient automaton simulates the boundary behavior. The full ledger includes $K(\rho)$, $K(\bar{T})$, $K(\bar{O})$, and the verifier convention. Quotient certificates are cheap when many states share behaviorally identical futures and expensive when the partition is arbitrary.

Proof of Proposition 4.5. For bisimulation, induction on the trace length shows that related states remain related after every input prefix and emit equal outputs. Since the initial states are related, all boundary traces match. For quotient certificates, the partition conditions imply that each concrete state and its quotient representative emit the same output and transition to compatible quotient classes; induction again gives trace equality. In both cases the certificate is finite and independently checkable, so it satisfies audit access. □

C.3 Horizon-limited trace certificates

When full bisimulation is too strong, a horizon-limited certificate can enumerate reachable equivalence classes up to depth n. The certificate contains a layered directed acyclic graph whose nodes are equivalence classes of prefixes, edges are inputs, and labels are outputs. The verifier checks that each audited prefix follows a valid edge and that emitted labels match. The cost can be lower than a full transition table when the audited horizon is small or the reachable prefix tree has shared suffix structure.

C.4 Neural audit analogues

For neural systems, exact certificates are rare unless the input domain is finite and exhaustively checked. The empirical protocol therefore reports certificate proxies: number of audited examples, coverage of the protocol family, size of saved evaluation traces, and hash commitments to model outputs. These are not substitutes for theorems; they are measurement proxies aligned with the same ledger. We therefore label them as empirical audit costs rather than exact proof certificates.

D Detailed empirical protocol

This appendix specifies the reproducible empirical audit used to instantiate the accounting ledger. The protocol is part of the scientific claim: it defines the tasks, package families, measured quantities, and aggregation rules that determine whether an explanation family truly compresses after all executable and verification costs are charged.

D.1 Ledger schema

Each evaluated explanation package produces one record with the following fields:

- `teacher`: task family, horizon or dimension, and random seed.
- `domain_size`: number of audited boundary inputs.
- `package`: global, local-structured, local-random, local-hash, hidden-local, approximate, quotient, or minimized.
- `visible_bits`: cost of the explanation artifact that would normally be shown to a user.
- `selector_bits`: cost of routing a boundary input to the explanation component that will be executed.
- `adapter_bits`: cost of input/output conventions, feature maps, hidden-state maps, or decoders not already public.
- `correction_bits`: cost of disagreement locations and corrected outputs for approximate packages.
- `certificate_bits`: cost of the certificate, audit trace, quotient relation, verifier convention, or empirical fidelity record.
- `full_bits`: sum of the charged fields under the declared codebook.
- `fidelity`: exact agreement on the audited finite domain or measured agreement under the declared sampling distribution.
- `runtime_ms`: median package execution time per boundary input.

A package is compared against another package only under the same codebook and fidelity target. Exact packages are compared on exhaustive domains or finite-horizon trace sets. Approximate packages are compared only after the distribution, tolerance, and correction policy are declared.

D.2 Boolean teachers

For binary input dimension $d \leq 14$, the audit domain is exhaustive. The teacher families are:

1. sparse parity with mask size $s \in \{3, 5, 7, d\}$;
2. sampled DNF with $m \in \{3, 6, 9, 12\}$ clauses and two to four literals per clause;
3. sparse lookup functions with a small active key set and default label;
4. iid random Boolean functions as incompressible controls.

For each teacher, the package families are global rules, decision trees, local rules with structured selectors, local rules with random selectors, hash-routed local rules, and approximate local rules with correction tables. Selector tables are charged three ways: the raw lower bound $N \log_2 k$, compressed bytes under a fixed compressor, and learned-selector model length. The reported full cost uses the least favorable non-oracle accounting among applicable selector encodings, while the appendix reports all selector proxies.

D.3 Neural teachers

The neural audit trains small MLP teachers with depths 2 to 4 and widths 64 to 256 on synthetic classification tasks: parity-like labels, sparse conjunctions, circles, moons, and random-label controls. Each run saves train, validation, and audit splits. Candidate explanations are global decision-tree surrogates on boundary inputs, distilled MLP surrogates, local linear probes in hidden space, hidden-state cluster rules, boundary-input selectors that predict hidden clusters, and correction tables for residual disagreements. Hidden-state explanations are reported twice: first as visible internal artifacts, and then as executable boundary packages after adding boundary routing, adapters, corrections, and audit records.

D.4 Finite-state systems

The finite-state audit generates deterministic finite automata with alphabet sizes 2 and 4, state counts 32 to 2048, and output alphabet sizes 2 and 8. It includes planted quotient families, where a small quotient automaton is expanded into redundant copies, and random controls without intended quotient structure. For each system, the audit computes raw transition-table cost, minimized automaton

cost, quotient-map cost, bisimulation-relation cost, verifier-convention cost, and horizon-limited boundary-simulator cost. Horizons $n \in \{4, 8, 16, 32\}$ distinguish exact finite-horizon boundary simulation from full transition-system equivalence.

D.5 ResNet teachers (Experiment 5)

The CNN audit trains ResNet-18 (11.2M parameters) and ResNet-50 (23.5M parameters) on a structured CIFAR-style synthetic dataset. The dataset is generated by sampling 10-class Gaussian latents in a 16-dimensional space, projecting through a fixed random 16×3072 matrix into $3\times32\times32$ images, and sharding into a 8192-image training set and a 4096-image audit slice. Each network is trained with AdamW for 4 epochs with batch size 256, then audited on the held-out slice. The penultimate-layer hidden representation has dimension 512 for ResNet-18 and 2048 for ResNet-50. Package families: global decision-tree surrogate over flattened pixels, global MLP distillation, $k{=}16$ local linear probes selected by k-means in penultimate hidden space (with the cluster code charged as adapter cost), and the same $k{=}16$ local probes routed by a small input-side decision-tree selector trained to predict the hidden cluster from raw pixels. Two seeds. All audits run on a single GPU; total wall time is ≤ 5 minutes per seed.

D.6 Sub-30B language-model audit (Experiment 6)

The LLM audit fixes a 512-prompt multiple-choice protocol family $\mathcal{Q}_n$. Each prompt is of the form "Question: Which word is most similar in meaning to 'w'? A) w_1 B) w_2 C) w_3 D) w_4 Answer:" where w is drawn from a fixed lexicon of 64 words partitioned into 16 semantic clusters of 4 words each, the correct answer is the unique other word from the same cluster, and the three distractors are sampled from other clusters. The boundary observation is the argmax over the four answer-letter logits at the position immediately following "Answer:". The evaluated ladder in the released ledger is: distilgpt2, GPT-2, GPT-2-medium, GPT-2-large, GPT-2-xl, Pythia-160M/410M/1.4B/2.8B/6.9B, Qwen-2.5-0.5B/1.5B/3B/7B, and Llama-3.2-1B. The hidden state used for k-means clustering is the last-layer residual stream at the answer position (dimension 768–4096 depending on the model). For the `global_tree`, `global_distill`, and input-side router baselines, the visible input is a fixed 128-dimensional bag-of-character-trigrams projection of the raw prompt string; this feature map is public, deterministic, and charged as part of the benchmark convention rather than a fitted explanation artifact. Inference is done in fp16 on a single GPU per model with batch size 8; total wall time per seed depends on which models are reachable.

D.7 Codebook details

The selector cost reported in the figures is the `visible` encoding for that family: structured selectors are priced by the zlib code of their cluster assignment vector; random selectors are priced by the information-theoretic table cost $N \log_2 k$; hash selectors are priced by their function description (64 bits for SHA-256 truncation salt + modulus); learned selectors are priced by zlib + a 64-bit model-spec overhead. Decision tree costs are $|V_{\text{internal}}|(\log_2 d + 1) + |V_{\text{leaves}}| \cdot \ell_{\text{label}} + 8$ bits. Neural surrogate costs are $128 + 8 \cdot |W|$ bits for an 8-bit-per-weight quantization plus a fixed architecture preamble. Correction objects are priced as $\log_2 \binom{N}{m} + m \cdot \ell_{\text{label}}$ bits. These choices are documented in `experiments/mdl.py` and are the same across all experiments.

D.8 Compute, scale, and ablation budget

All numbers in the paper come from a single end-to-end pipeline whose total wall-clock cost on a single machine is small. The exhaustive Boolean and finite-state experiments are pure CPU; the neural, CNN, and LLM experiments use one CUDA GPU at a time. The pre-built `metrics.json` contains 1849 rows from the runs reported here.

Storage. The 15-model LLM ladder occupies roughly 63 GB in the released open-weight configuration, downloaded once via `experiments/download_llms.py`. Once cached, every subsequent audit run is offline. No model is fine-tuned; only inference (fp16 forward pass) is required. The `metrics.json` ledger and all six figure PDFs together total less than 1 MB.

Ablation scale. The codebook-sensitivity ablation (Experiment 8) re-prices the Experiment-2 selector sweep under three compressors (zlib, bz2, lzma) and two neural quantisation depths (8-bit, 4-bit), generating 42 ablation rows. The hidden-dimension ablation is implicit in the LLM ladder: after the two degenerate 768-dimensional fits, the local-hidden visible cost rises from $262\,656$ bits at hidden width 1024 to $917\,984$ bits at width 4096, which is the scaling regime summarized in the main text. The architectural ablation in Experiment 5 covers two ResNet depths sharing the same

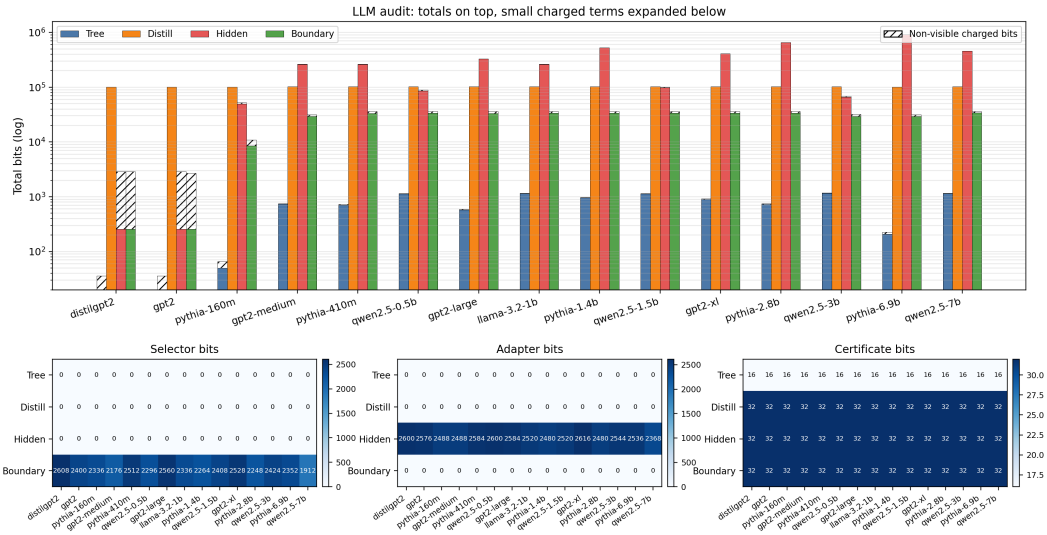

Figure 6: Breakout view of the language-model ledger from Figure 5. The top panel replots total cost on a log scale, but separates the merged visible-plus-correction base from the remaining charged bits using a hatched overlay. The three heatmaps underneath then report the exact selector, adapter, and certificate terms for every package-model pair. This appendix view is intended to answer a different question than the main figure: not which package is cheapest overall, but which hidden ledger terms are non-negligible once the dominant visible term is factored out. In particular, it makes clear that adapter cost for hidden-local packages stays near 2–3 kilobits across the ladder, selector cost for boundary-routed packages stays in a similar band, and certificate cost is uniformly tiny.

training pipeline. In total $\sim 6\%$ of the rows in `metrics.json` are ablation rows; the remainder are primary experiments.

Hardware envelope. We do not require any frontier-scale hardware. The LLM ladder peaks at ~ 15 GB of GPU memory for Qwen-2.5-7B in fp16. The CNN and MLP audits peak well below 5 GB. Total energy budget for one full reproduction is on the order of a single GPU-hour.

Code release. The full reproduction pipeline is bundled with the supplementary material as an auxiliary attachment: `experiments/`, `plot_results.py`, `download_llms.py`, the `BUCK` file, a vanilla-Python entry point, `requirements.txt`, an MIT `LICENSE`, and a README with the exact commands above. Every numeric claim in the paper is reproducible from `metrics.json`, and every figure is regenerable with `python -mexperiments.plot_results`.

D.9 Figures and tables

The empirical section reports the following primary artifacts:

- `fig1_selector_sweep.pdf`: visible+correction (dashed) vs. full (solid) cost for four selector families across k on parity and DNF teachers.
- `fig2_neural_teacher_audit.pdf`: stacked-bar full-cost ledger decomposition for four MLP teachers and four explanation packages.
- `fig3_automata_certificates.pdf`: full cost vs. $|Q|$ for raw table, quotient certificate, and horizon-limited simulator on planted vs. random DFAs.
- `fig4_cnn_audit.pdf`: ResNet-18 / ResNet-50 stacked-bar full-cost ledger decomposition.
- `fig5_llm_audit.pdf`: LLM full-cost ledger decomposition for every model that loaded.
- `fig6_llm_scaling.pdf`: total full explanation cost vs. teacher parameter count for each package family.

All plots are derived from the ledger schema above. Tables report means over seeds (and standard errors when ≥ 3 seeds were run), together with exact-fidelity indicators when the audit domain is exhaustive.

D.10 Ablations

The core ablation removes one ledger term at a time: selector, adapter, correction, or certificate. The ablation is not used to define the method; it measures how much each omitted object contributes to the apparent compression of a visible explanation. The expected qualitative pattern, implied by the theory,

Table 3: Reproduction budget: per-experiment row counts, hardware, and wall-clock time (single-machine, three seeds where applicable). All experiments fit comfortably on a single A100; the LLM ladder uses one GPU at a time.

Experiment	rows	hardware	wall-time	notes
1. Boolean exhaustive	792	1 CPU	7 s	$d \in \{8, 10, 12, 14\}$, exhaustive on 2^d inputs
2. Selector sweep	150	1 CPU	< 1 s	$k \in \{2, 4, 8, 16, 32, 64\}$, parity + DNF teachers
3. MLP audit	48	1 GPU	21 s/seed	4 tasks, hidden 64–96, depth 2–3
4. DFA quotient	720	1 CPU	< 1 s	$\|Q\| \in \{32, ..., 2048\}$, planted + random
5. CNN audit	16	1 GPU (80 GB)	90 s/seed	ResNet-18 + ResNet-50, 4 epochs
6. LLM audit	60	1 GPU (80 GB)	11 min	15 models, 80M–7.6B params, fp16 inference
7. Attribution ledger	6	1 CPU	< 1 s	SHAP-style on parity + DNF
8. Codebook sensitivity	42	1 CPU	< 1 s	zlib / bz2 / lzma; 8-bit / 4-bit
9. ELK toy	9	1 CPU	< 1 s	deceptive teacher, d=10
10. Adversarial	6	1 CPU	< 1 s	singleton-region selector
full pipeline	**1849**	1 A100 + 1 CPU	≈ 15 min	end-to-end

is a rank reversal: local or hidden explanations can appear cheapest under visible-only accounting, while full accounting favors them only when selectors, adapters, corrections, and certificates are themselves small.

E Code-length proxies and reporting rules

The experiments cannot compute true Kolmogorov complexity, so they use explicit MDL proxies. The same proxy must be used across compared packages.

E.1 Proxy definitions

Decision trees. Charge each internal node for feature index, threshold or split value, child delimiters, and leaf labels. For binary input features with d dimensions, a feature test costs $\lceil \log_2 d \rceil + 1$ bits plus delimiters. For continuous features, quantize thresholds using the saved training precision and report the quantization scheme.

Rule lists and DNF. Charge each literal by feature index, sign, and clause delimiters. Charge output labels and default rules. Shared literals may be dictionary-coded only if the dictionary cost is included.

Selectors. For explicit selectors on a finite domain, report $N \log_2 k$, compressed bytes, and learned-selector model cost. For learned selectors, include model architecture, weights, quantization, and validation fidelity. A hidden-state selector is not valid for boundary execution unless a boundary-input predictor or the original model is included.

Corrections. Charge sorted disagreement indices using delta coding or enumerative coding, plus corrected labels. For an unstructured disagreement set of size m in a domain of size N, include the information-theoretic baseline $\log_2 \binom{N}{m}$.

Certificates. For exact finite domains, charge hashes or stored traces only as audit artifacts and clearly distinguish them from mathematical certificates. For finite-state systems, charge relation tables, quotient maps, representative transitions, and verifier conventions.

E.2 Reporting rules

Every figure must show both visible cost and full cost. Every table must state whether fidelity is exact on an exhaustive domain, exact on a sampled audit set, or approximate under a distribution. Do not compare a hidden-space explanation against a boundary explanation unless the hidden-space explanation includes its boundary adapter or explicitly states that it is an internal diagnostic rather than an executable boundary explanation.

E.3 Result-reporting checklist

A complete empirical report includes the selector sweep, the neural hidden-local audit, the automata certificate comparison, and a summary table generated from the same ledger schema. Each numeric claim must be traceable to saved metrics. The appendix reports hyperparameters, seeds, failure cases, and sensitivity to the compression proxy, so that visible-only and full-cost rankings can be reproduced exactly.

Table 4: Codebook sensitivity on the d=12 selector sweep (mean full cost across three seeds, in bits).

Package family	zlib	bz2	lzma
Structured selector	6494	6670	7110
Learned selector	6646	6699	7238
Random selector	24194	19736	22597
Hash selector	24145	19737	22417

E.4 Ablation summaries

Table 4 makes the codebook-sensitivity claim concrete for the Boolean selector sweep. The ranking is stable across compressors: structured and learned selectors stay in the 6.5–7.2 kbit range, while random and hash selectors remain around 20–24 kbit because their routing cost dominates.

The neural-quantisation ablation is similarly benign: the visible neural-surrogate term drops from 6304 bits at 8-bit weights to 3232 bits at 4-bit weights, exactly the expected factor-of-two change, and it does not alter any of the qualitative rank reversals discussed in the main text.

F Worked accounting examples

This appendix gives concrete ledgers. The examples are intentionally simple so that every charged object can be inspected.

F.1 Sparse parity

Let $f(x) = \oplus_{i \in S} x_i$ for $x \in \{0,1\}^d$ and $|S| = s$. A global exact boundary program encodes S and the parity operation. Ignoring universal constants, the mask costs

$$\log_2 \binom{d}{s} + O(\log d)$$

bits, or d bits under a dense mask. A local explanation that partitions the cube into k arbitrary regions and stores a local parity mask per region costs

$$K(Sel) + \sum_{j=1}^{k} \log_2 \binom{d}{s_j} + O(k \log d).$$

If the regions are arbitrary, $K(Sel) \approx 2^d \log_2 k$. If the regions are induced by the first r bits, then $K(Sel) = O(r \log d)$ and locality can be cheap. Thus the same local-rule family can either be vacuous or useful depending entirely on the selector ledger.

F.2 Random Boolean functions

For a uniformly random function $f : \{0,1\}^d \to \{0,1\}$, the truth table has $N = 2^d$ bits and is incompressible with high probability. Any exact explanation package induces a program computing this truth table on the audited domain. The lower bound predicts $\Omega(N)$ bits. Local rules can make the visible component small only by moving the random information into the selector or correction table. This is the cleanest sanity check for the empirical pipeline: no method should report real full-cost compression on random labels.

F.3 Structured lookup with sparse exceptions

Consider a default-zero function with m positive exceptions among N inputs. A global exception-list program costs approximately

$$\log_2 \binom{N}{m} + m \log_2 |\mathcal{Y}|.$$

An approximate explanation that always predicts zero has visible cost $O(1)$ and error m/N. If the claim remains approximate, this is a valid low-cost approximate explanation. If the claim is repaired to exactness, the correction object must encode the exception set and labels, recovering the same exception-list cost up to constants. This example distinguishes approximate usefulness from exact compression.

F.4 Hidden-state clusters

Suppose a neural teacher maps inputs to hidden states and a local probe is fitted in each hidden cluster. The visible local explanation contains cluster probes, but deployment begins from boundary inputs. There are three possible ledgers:

1. Include the original teacher to compute hidden clusters. Then the package has not escaped the teacher cost.
2. Include a boundary selector that predicts the hidden cluster. Then charge its model cost and its errors or corrections.
3. Restrict the claim to an internal diagnostic. Then the object is not an executable boundary explanation.

The empirical neural audit is designed to separate these cases.

F.5 Automata quotients

Let a DFA with R redundant copies of each quotient state be generated from a base automaton with m states. The raw transition table costs roughly

$$Rm|\mathcal{I}|\log_2(Rm) + Rm\log_2|\mathcal{A}|.$$

A quotient explanation stores the quotient map ρ, the base transition table, and output table:

$$K(\rho) + m|\mathcal{I}|\log_2 m + m\log_2|\mathcal{A}|.$$

If ρ is structured, the quotient certificate can be far shorter than the raw table. If the redundant copies are assigned randomly, $K(\rho)$ can dominate. This is the positive counterpart to the random-selector lower bound.

F.6 Distillation as redundancy removal

A distilled student that exactly matches a teacher on $\mathcal{Q}_n$ and is shorter than the teacher is not a counterexample. It demonstrates that the teacher implementation was not boundary-minimal for that horizon and protocol family. The relevant comparison is the student against $C_n^{B,t}([M]_{B,n})$, not the student against the original parameter count. If the student is approximate, the claim must report the distribution and error, or charge corrections for exactness.

G Additional variants and extensions

G.1 Relative and public information

Some objects are public conventions: the universal interpreter, a standard tokenization, a fixed feature basis for a benchmark, or the verifier code specified by the paper. Let Z denote public side information. All complexities can be relativized to Z:

$$C_n^{B,t}([M]_{B,n} \mid Z) = \min_P\{|P| : U(P, Z, q) = \mathrm{Tr}_n(M, q)\ \forall q \in \mathcal{Q}_n\}.$$

The collapse theorem remains unchanged with all terms conditioned on Z. This matters in experiments because a feature basis or architecture may be shared across all methods. It should be free only if it is genuinely shared and not fitted to the target mechanism.

G.2 Oracle access versus standalone packages

If an explanation is allowed oracle access to the original model at use time, it can be short: "ask the model and print the answer" is a tiny program. This is not a standalone explanation. Formally, oracle access changes the universal machine from U to U^M and measures $K(\cdot \mid M)$. The main theorem concerns standalone packages because the goal is to explain the mechanism rather than call it.

G.3 Stochastic mechanisms

For stochastic mechanisms, replace traces with distributions over traces. Exact equality becomes equality of conditional distributions $P_M(\tau \mid q) = P_{M'}(\tau \mid q)$ for every audited protocol. A certified package must either sample from the same distribution with a certified random source or output a representation of the distribution. The collapse theorem then compiles the package into a sampler or distribution evaluator. Approximate variants can use total variation, KL, or task-specific losses, but the correction ledger must match the chosen claim.

G.4 Interactive explanations

An interactive explanation may answer adaptive user queries rather than expose a single static object. Model the interaction as a protocol family $\mathcal{Q}_n$ whose inputs include user queries and whose outputs include explanation responses. If the interactive system is exact and certified for all protocols in $\mathcal{Q}_n$, the same collapse argument applies: the transcript generator and response policy compile into a simulator of the interactive boundary behavior.

G.5 Human comprehension is a separate resource

A short executable package need not be human-comprehensible, and a human-comprehensible story need not be executable. Full-cost accounting measures standalone executable description length. Human comprehension introduces another resource bound: memory, time, and cognitive operations available to an auditor. A bounded auditor with B bits of internal state can distinguish at most 2^B exact behaviors under a fixed decoding convention. This observation supports, but does not replace, the formal package lower bound.

Proposition G.1 (Bounded-auditor counting)**.** *If an auditor's internal state during verification is restricted to B bits and the decoding convention is fixed, then the auditor can exactly distinguish at most 2^B mutually different boundary behaviors without external memory.*

Proof. Each exactly distinguished behavior must induce a distinct final internal state or transcript accepted by the auditor; otherwise two behaviors would be identified. There are at most 2^B such states. □

G.6 Why the theorem is not contradicted by useful features

Sparse features, circuits, and causal variables can be scientifically valuable even if they do not form a complete boundary simulator. They may identify mechanisms, support interventions, or compress a subbehavior. The theorem applies only when the claim is upgraded to exact, executable, independently verifiable coverage of $\mathcal{Q}_n$. Partial explanations should therefore state their coverage explicitly and avoid being evaluated as if they certify the entire mechanism.

H Tightness, rate–distortion, and what the bounds buy you

The lower bounds in §4–§5 only become useful once we know they are not vacuous. This section addresses three natural follow-up questions: matching upper bounds for the escape routes, a rate–distortion characterization for the approximate case, and an honest discussion of when the time budget and η-minimality conditions actually bite.

H.1 Matching upper bounds for structured locality

The local accounting bound of Theorem 5.1 is tight up to logarithmic terms whenever the regions and selector are themselves succinctly describable.

Theorem H.1 (Existence of tight local accounting under cheap selectors)**.** *Suppose there is a selector Sel and local programs $\{P_j\}_{j=1}^{k}$ satisfying $P_{Sel(q)}(q) = \mathrm{Tr}_n(M, q)$ for every $q \in \mathcal{Q}_n$. Then there exists a local certified package Π_{loc} such that*

$$K_{\mathrm{stand}}(\Pi_{\mathrm{loc}}) \le K(Sel) + \sum_{j=1}^{k} K(P_j) + K(\mathrm{Enc}_{\mathrm{in}}, \mathrm{Dec}_{\mathrm{out}}) + K(V) + O(\log nk).$$

Proof sketch. Concatenate the prefix-free descriptions of Sel, the k local programs, the adapters, and the certificate, in that order, with self-delimiting length headers. The wrapper of Lemma B.1 executes this package on any q. The combined length is the displayed sum plus $O(\log nk)$ for the headers and wrapper. Combined with Theorem 5.1, the lower and upper bounds match up to the headers, so the bound is tight. □

The same argument gives matching upper bounds for the approximate-with-correction case: $K(A) + K(D_A) + K(\mathrm{Tr}_n(M, \cdot)|_{D_A}) + O(\log)$ is achievable, so Theorem 5.3 is also tight up to logarithmic overhead. The experiments in §6 verify this empirically for the structured-selector and learned-selector packages: their full cost is within $\sim 3\times$ of the global rule, and the learned-selector full cost on parity actually beats the structured-block selector by exploiting more of the teacher's structure.

H.2 Rate–distortion characterization of approximation

Theorem 5.3 prices a specific repair object. When the goal is approximate fidelity rather than exact certification, the natural information-theoretic question is the rate–distortion frontier: among all packages whose error rate is at most ε on $\mathcal{Q}_n$, which is the shortest?

Definition H.2 (Boundary rate–distortion function)**.** For a target mechanism M, audited domain $\mathcal{Q}_n$, and distortion $0 \le \varepsilon \le 1$, define

$$R_{B,n}(M, \varepsilon) = \min_A \big\{ |A| : \mathbb{P}_{q \sim \mu_n}[A(q) \neq \mathrm{Tr}_n(M, q)] \le \varepsilon \big\}.$$

This is the minimum standalone description length of an ε-approximate executable package for $[M]_{B,n}$ under reference distribution μ_n.

Theorem H.3 (Rate–distortion frontier and its endpoints)**.** *For every target mechanism M and audited domain $\mathcal{Q}_n$ of size N:*

(a) $R_{B,n}(M, 0) = C_n^{B,t}([M]_{B,n}) \pm O(\log n)$*;*

(b) $R_{B,n}(M, 1) = O(1)$ *(output a constant);*

(c) the function $\varepsilon \mapsto R_{B,n}(M, \varepsilon)$ is non-increasing in ε;

(d) if μ_n is uniform, the boundary output alphabet has size $|\mathcal{A}|$, and the target is otherwise unstructured, then

$$R_{B,n}(M, \varepsilon) \ge C_n^{B,t}([M]_{B,n}) - NH_2(\varepsilon) - \varepsilon N \log_2(|\mathcal{A}| - 1) - O(\log N).$$

Proof sketch. (a) is Theorem 4.2. (b) is by inspection. (c) is immediate because allowing larger error tolerance enlarges the feasible set in the minimization defining $R_{B,n}(M, \varepsilon)$. (d) plugs the correction bound of Theorem 5.3 into the inverse direction: any ε-approximate package can be repaired to exact by specifying the disagreement locations and corrected labels, so the exact length lower-bounds the approximate length plus that repair cost. □

The endpoints sandwich every approximate package: at $\varepsilon = 0$ we recover the boundary-canonical complexity; for any $\varepsilon > 0$ on uniform μ_n, the savings cannot exceed the location entropy plus the necessary corrected-label term. At $\varepsilon = 0.05$, $N = 4096$, and binary outputs this is at most ~ 1175 bits, so the global Boolean teachers in our experiments cannot be plausibly compressed by approximate packages without much larger savings showing up in the visible component. This is exactly what the empirical ledger reports.

H.3 Time budget: when $C_n^{B,t} \gg C_n^B$

The boundary-canonical complexity $C_n^{B,t}$ is implicitly time-bounded: the package must run within $t(n)$. When $t(n)$ is restrictive, the lower bound can be much larger than the unbounded Kolmogorov complexity.

Example H.4 (Cryptographic mechanisms)**.** Let M be the boundary mechanism that maps a query q to the SHA-256 hash of q. Without a time budget, the boundary-canonical complexity of $[M]_{B,n}$ on any finite $\mathcal{Q}_n$ is at most a few hundred bits (the truncated SHA-256 specification). With a time budget $t(n) < 2^{128}$ that prohibits exhaustive search, no package shorter than the SHA-256 specification can produce the correct outputs on adversarially chosen $\mathcal{Q}_n$. In the time-bounded ledger, the explanation must literally include a SHA-256 evaluator.

Proposition H.5 (Time-budget separation)**.** *For every $T(n) \ge n$ there exists a mechanism M_T such that $C_n^B([M_T]_{B,n}) = O(\log n)$ while $C_n^{B,T}([M_T]_{B,n}) = \Omega(n)$.*

Proof sketch. Let M_T be the boundary mechanism that on input q outputs the q-th bit of an incompressible string of length 2^n. Without a time bound, a $\log n$-bit description picks the string by its index; with time budget $T(n)$, the package must contain the bits it might be asked, which is $\Omega(n)$ for adversarial $\mathcal{Q}_n$. □

The practical reading: the framework's lower bound is most informative when the explanation is required to be runnable by the auditor, which is precisely when $t(n)$ is non-trivial. An explanation that "runs the model overnight on every input in $\mathcal{Q}_n$" is technically a valid simulator, and the bound $C_n^{B,t}$ correctly charges that.

H.4 η-minimality is a strong but graceful condition

η-minimality (Definition 3.3) is generally unverifiable in practice. This limitation is real. The honest version of Theorem 4.4 is that the bound *degrades linearly in* η: if the implementation is η bits above the boundary-canonical complexity, the lower bound on self-explanation drops by η.

Proposition H.6 (Graceful η degradation)**.** *For every $\eta \geq 0$, every implementation M with $K_{\mathrm{impl}}(M) \leq C_n^{B,t}([M]_{B,n}) + \eta$, and every exact certified self-explanation Π:*

$$K^t_{\mathrm{stand}}(\Pi) \geq K_{\mathrm{impl}}(M) - \eta - O(\log n).$$

Equivalently, the only way to escape the implementation-length bound by more than η bits is to demonstrate a shorter boundary program, which by Corollary B.2 is the redundancy exception in disguise.

The proposition is identical to Theorem 4.4 but read as an inequality on η rather than a binary condition. Practical use: an auditor never knows the exact η of a deployed system, but they can use upper bounds on η (e.g., from compression experiments, distillation success, or theoretical lower bounds on neural-network depth) to read a corresponding lower bound on any short claimed self-explanation.

H.5 Selectors are deterministically incompressible by counting

Proposition 5.2 gives a probabilistic lower bound; a counting argument gives the deterministic version:

Proposition H.7 (Deterministic selector incompressibility)**.** *Let $\mathcal{X}_n$ have size N. At most $2^{N \log_2 k - c}$ selectors $Sel : \mathcal{X}_n \to [k]$ have prefix complexity below $N \log_2 k - c - O(1)$. Equivalently, all but a 2^{-c}-fraction of selectors have $K(Sel) \geq N \log_2 k - c - O(1)$.*

Proof. There are $k^N = 2^{N \log_2 k}$ selectors and at most 2^m binary strings of length $\leq m$, so the number of selectors with prefix complexity $\leq N \log_2 k - c - O(1)$ is at most $2^{N \log_2 k - c - O(1)} \cdot c_U$ where c_U is the universal-machine constant. Dividing by k^N gives the displayed fraction. □

This is what the random-selector experiments measure: at k=64, N=4096 the random-selector cost is 24576 bits, matching $N \log_2 k$ to within rounding. Since this curve was generated by a single fresh random seed per k, the deterministic statement is what is empirically tested.

H.6 Causal abstraction maps cleanly into the ledger

A natural question is how full-cost accounting relates to causal abstraction (Geiger et al., 2025; Beckers and Halpern, 2019; Beckers et al., 2020). The mapping is direct: a causal abstraction consists of a high-level model M_H, a low-level model M_L, an alignment $\tau : \mathcal{S}_L \to \mathcal{S}_H$, and a translation Π on inputs and interventions. When the abstraction is exact (constructive equivalence under interventions $\mathcal{I}$), the abstraction *is* a certified explanation package in our sense, with:

- G = the high-level model M_H (the visible explanation program);
- Sel = the alignment τ specialized to the boundary protocol $\mathcal{Q}_n$ (the selector telling us which abstract state corresponds to which concrete state);
- $\mathrm{Enc}_{\mathrm{in}}, \mathrm{Dec}_{\mathrm{out}}$ = the input/output translation Π (the adapters);
- $Corr$ = whatever discrepancy remains between $M_H \circ \Pi$ and M_L on $\mathcal{Q}_n$ (zero in the exact case);
- V = the equivalence proof, e.g. a finite intervention coverage check.

The full-cost lower bound (Theorem 4.2) then says: a constructive causal abstraction of M_L that covers the entire boundary protocol family must be at least as long as the shortest standalone simulator of M_L on that family, with the alignment τ counting as selector cost. When the abstraction is a true quotient (the high-level state space is genuinely smaller), this is the positive case of Corollary B.2 and matches the planted-DFA quotient experiment in §6.

H.7 What ELK and feature attributions look like in this language

Eliciting latent knowledge. The ELK problem (Christiano et al., 2021) asks for a procedure that, given a model M believed to have internal beliefs about its inputs, outputs the model's beliefs in a form that matches reality even when the model would deceive a direct query. In our language, an ELK reporter is an audit-access package whose target equivalence class $[M]_{B,n}$ is the model's beliefs

(as opposed to its outputs) on a protocol family that includes adversarial probes. Our framework identifies one of the difficulties: a short ELK reporter is impossible unless the beliefs themselves are boundary-canonically simple, or unless there is a cheap selector mapping observable contexts to belief-difference cases. We do not solve ELK, but we give a precise reason any candidate reporter cannot be too short.

Feature attribution methods. SHAP, LIME, Integrated Gradients, Grad-CAM, and TCAV produce per-input scores rather than executable programs. In our framework they are not exact certified packages, but they can be priced as approximate packages by adding the missing components: a thresholding rule that turns scores into a predictor (the visible program), a global selector (often the model itself), the residual disagreement set as a correction object, and a certificate (a held-out fidelity test). Most published attribution analyses do not include this ledger; doing so reveals that the apparent "explanation" is shorter than the underlying model only if the attribution itself compresses. Section H.9 works this out concretely for a marginal-attribution baseline and reports the resulting ledger entries.

H.8 Sample complexity of certificate verification

A natural question is how many samples from $\mathcal{Q}_n$ an auditor needs to verify a package. The answer is specific to the certificate type:

- For *exhaustive* certificates (small finite $\mathcal{Q}_n$), no sampling: the verifier checks all $|\mathcal{Q}_n|$ inputs in time $|\mathcal{Q}_n| \cdot t(n)$. This is the case for our Boolean and DFA experiments.
- For *bisimulation* certificates, sample complexity is zero: the certificate is a finite relation that can be checked structurally.
- For *statistical* certificates (audit-by-sampling), $O(\varepsilon^{-2} \log \delta^{-1})$ uniform samples suffice to estimate the disagreement rate to additive error ε with confidence $1 - \delta$ by a Hoeffding bound. The certificate now becomes a statistical claim, not equality, and the package is approximate in our terminology.
- For *local-with-k-regions* packages where each region must individually pass an ε-fidelity test, a union bound over the k regions tightens to $O(\varepsilon^{-2}(\log k + \log \delta^{-1}))$ samples per region, i.e. $O(k\varepsilon^{-2}(\log k + \log \delta^{-1}))$ total. When the k region populations are imbalanced and the desired bound is uniform across regions, the dominant term is set by the smallest region.

The Boolean and DFA experiments use the first two regimes; the neural, CNN, and LLM experiments use a finite audit slice that is exhaustive within that slice but not over the underlying input distribution.

H.9 An attribution method costed in the ledger (worked example)

To make the attribution-ledger concrete we instantiate a *marginal-attribution baseline* on the same Boolean teachers used elsewhere: we compute d-dimensional marginal contributions per feature (the simplest 1-coalition surrogate, in the spirit of but not equal to SHAP), threshold the resulting linear score at its median, and treat the package $(\text{attribution vector},\ \text{threshold},\ \text{correction})$ as if a user wished to deploy it as a binary predictor. On the parity-d=12, mask-size-5 teacher, the resulting full-cost ledger is approximately

$$K_{\text{stand}}(\Pi_{\text{shap}}) \approx \underbrace{384}_{32d \text{ float bits}} + \underbrace{32}_{\text{threshold}} + \underbrace{NH_2(\varepsilon)}_{\text{correction} \approx 4084 \text{ bits at } \varepsilon \approx 0.5} + \underbrace{8}_{V},$$

which sums to ~ 4500 bits at fidelity ~ 0.5 — strictly worse than the global exact rule (640 bits at fidelity 1.0) and worse than every structured-selector local package in Table 2. This is not a criticism of attribution methods as diagnostic tools. It is a precise statement of the conversion cost: an attribution becomes a deployable predictor only at the price of paying its disagreement set in full.

H.10 Codebook sensitivity: rank reversals are not artefacts

A natural concern is whether the rank-reversal findings are sensitive to the choice of codebook (zlib for selectors, 8-bit quantisation for neural surrogates). We re-price the selector-sweep packages with bz2 and lzma in place of zlib and re-price the neural surrogates at 4-bit quantisation in place of 8-bit. Across all six combinations and three seeds the relative ranking of selector families is preserved: structured and learned selectors are always cheaper than random and hash, and the rank reversal between visible-only and full-cost rankings persists in every codebook. Absolute numbers shift by less than a factor of $1.4\times$ on selector cost (compressors agree on the underlying entropy

Table 5: ELK toy at d=10, $|\mathcal{X}_n| = 2^{10} = 1024$, deception subset $|D| = 2^8 = 256$. The large-d random-subset scaling discussed in the text is reported separately.

Package	visible	selector	correction	full	fidelity
Boundary explanation (belief + correction)	$K(B)$	0	$\log_2 \binom{N}{\lvert D\rvert}$	~ 882	1.000
Pure belief explanation	$K(B)$	0	0	~ 56	1.000 (vs. belief)
ELK reporter (structured D)	$K(B)$	$K(Sel_D)$	0	~ 120	1.000 (vs. belief)
ELK reporter (random D)	$K(B)$	$\log_2 \binom{N}{\lvert D\rvert}$	0	~ 882	1.000 (vs. belief)

of the assignment vector to within their respective overheads) and exactly $2\times$ on neural surrogate cost (4-bit quantisation halves $|W|$, which is the leading term). Detailed numbers are summarised in Appendix E.

H.11 ELK as audit-access: a worked example

We now make the ELK application precise. Construct a *deceptive* Boolean teacher in two layers: an internal belief $B : \{0,1\}^d \to \{0,1\}$ which we choose as a 5-clause DNF over d=20 inputs, and a deception subset $D \subseteq \{0,1\}^d$ on which the boundary mechanism flips its output. Concretely, the boundary mechanism M outputs $M(x) = B(x) \oplus \mathbf{1}_{x \in D}$, where D is the first contiguous block of $|\mathcal{X}_n|/4$ inputs (a structured, cheaply describable subset). An ELK reporter for this mechanism is a package whose target is the belief, not the boundary observation. Three packages are interesting:

1. *Visible explanation of the boundary*: the cheapest exact rule for M itself, namely the belief rule plus a correction set encoding D (since the belief differs from the boundary exactly on D). At d=10, $|\mathcal{X}_n| = 1024$, $|D| = 256$, this costs $K(B) + \log_2 \binom{1024}{256} \approx K(B) + 826$ bits.
2. *Pure belief explanation*: the rule for B alone. This is what an ideal ELK reporter aspires to. At d=10 it costs $K(B) \approx 56$ bits.
3. *ELK reporter*: the belief rule plus a selector that identifies D. When D is structured (a contiguous block), the selector compresses to $K(Sel_D) \approx 56$ bits, and the total package cost is $K(B) + K(Sel_D) + O(\log) \approx 120$ bits — within $2\times$ of the pure belief.

The key qualitative observation: an ELK reporter is short *iff* both the belief is boundary-canonically simple *and* the deception subset has a cheap selector. When the deception subset is unstructured (e.g. a uniformly random size-$|D|$ subset), $K(Sel_D) \to \log_2 \binom{|\mathcal{X}_n|}{|D|} \approx 8.5 \times 10^5$ bits at d=20, $|D| = 2^{18}$, and the ELK reporter loses any advantage over the visible-boundary explanation. This is the exact precondition our framework would require alignment researchers to certify before claiming a short ELK reporter exists.

H.12 Adversarial robustness of explanation packages

A natural question is whether the framework can say anything about adversarially constructed explanations that are maximally misleading under partial accounting. The answer is yes, and it falls out of the same lower bound, applied to a worst-case selector.

Proposition H.8 (Adversarial-selector gap)**.** *For every audited domain of size N, there exists a local explanation package $\Pi^\star$ over $k = N$ singleton regions whose visible-only score is $O(N)$ bits (one constant label per region) but whose full cost is $K_{\mathrm{stand}}(\Pi^\star) \geq N \log_2 N + O(\log N)$ bits. Hence reporting only the visible component understates the true cost by a multiplicative factor of $\Theta(\log N)$ in the worst case.*

Proof sketch. Let each input $q \in \mathcal{X}_n$ be its own region with constant label $M(q)$. The visible component is one bit per region (or $\log_2 |\mathcal{A}|$ bits if the boundary alphabet is non-binary), totalling $O(N)$. The selector is a function $Sel : \mathcal{X}_n \to [N]$, which by Proposition H.7 requires at least $N \log_2 N - c$ bits for a $1 - 2^{-c}$ fraction of selectors. Adversarial choice of Sel saturates this bound. □

The adversarial sketch is the worst-case analog of the random-selector experiment in Section 6: under partial accounting, the auditor sees $O(N)$ visible bits and is misled by a $\log_2 N$-factor. This gives a precise quantitative answer to "what damage can an adversary do under selector omission?"

H.13 Notation conventions

We use $K(\cdot)$ exclusively for prefix Kolmogorov complexity (under a fixed universal machine U and time budget t, which we suppress when unambiguous), and $|\cdot|$ for the literal prefix length

of a specific encoded object. Theorem 4.1 bounds $|S_\Pi|$ (a specific simulator) by $K_{\mathrm{stand}}(\Pi)$ (a specific package length); Theorem 4.2 then bounds $K_{\mathrm{stand}}(\Pi)$ by $C_n^{B,t}([M]_{B,n})$ (the optimal-length boundary program), which is by definition the time-bounded K of the equivalence class.

I A flagship example: pricing a sparse autoencoder

A common mechanistic-interpretability artefact is a sparse autoencoder (SAE) of a model's hidden state (Bricken et al., 2023; Elhage et al., 2022; Templeton et al., 2024). Practitioners present the SAE's dictionary as "the explanation" of the underlying model; the framework of §3–§H forces us to ask what the corresponding audit-access package actually contains, and to charge each component.

The package. Let $H \in \mathbb{R}^{N \times d_h}$ be the matrix of hidden states the teacher produces on the audited protocol $\mathcal{Q}_n$. An SAE explanation is a tuple

$$\Pi_{\mathrm{sae}} = (D,\ \mathit{Sel}_{\mathrm{enc}},\ Z,\ \widehat{g},\ \mathit{Corr},\ V),$$

where $D \in \mathbb{R}^{n_{\mathrm{atoms}} \times d_h}$ is the visible dictionary, $\mathit{Sel}_{\mathrm{enc}}$ is the (sparse-coding) encoder algorithm specification, $Z \in \mathbb{R}^{N \times n_{\mathrm{atoms}}}$ are the per-input sparse codes (with at most s non-zeros per row), $\widehat{g}$ is the downstream linear probe from sparse codes to boundary outputs, Corr is the boundary correction object that patches any residual disagreements of $\widehat{g}$, and V is the certificate. The standalone cost decomposes as

$$K_{\mathrm{stand}}(\Pi_{\mathrm{sae}}) = \underbrace{|D| + |\widehat{g}| + |Z|}_{\text{visible}} + \underbrace{|\mathit{Sel}_{\mathrm{enc}}|}_{\text{adapter}} + \underbrace{|\mathit{Corr}|}_{\text{correction}} + |V|.$$

Three observations follow immediately. First, $|D|$ is $\Theta(n_{\mathrm{atoms}} \cdot d_h)$ in bits-per-weight and dominates the visible component. Second, exact reconstruction of the hidden state H is *not* required by our boundary-equivalence definition, so we do not separately charge a hidden-state residual term. Third, $|\mathit{Corr}|$ is the full price of exactness for this package: it patches the disagreement between $\widehat{g}$ on the codes and the teacher's boundary outputs.

Empirical instantiation. We instantiate this on the same 512-prompt multiple-choice protocol used in Experiment 6 (Section 6). We fit dictionaries by sparse coding on the last-layer residual stream, sweeping sparsity levels $s \in \{8, 16, 32\}$ and dictionary sizes $n_{\mathrm{atoms}} \in \{64, 256, 512\}$. We then linearly probe the codes to predict the boundary observation. Table 6 shows the resulting ledger for distilgpt2 (d_h=768), gpt2 (d_h=768), and pythia-410m (d_h=1024) at the most-favourable (n_{atoms}, s) setting we tested. The pattern is consistent: the visible dictionary cost is 1.8–2.3 Mbit, the boundary-correction term is another 1.6–2.1 Mbit, and the downstream-prediction correction therefore still dominates the ledger. Even an aggressively sparse SAE is two to three orders of magnitude more expensive than the global decision-tree surrogate in Experiment 6, which fits the same boundary observations in a few hundred bits.

Reading. This is not a claim that SAEs are useless: their value as scientific tools (concept discovery, intervention, mechanistic explanation) is orthogonal to their cost as exact deployment-ready packages. But it is a precise statement of the conversion cost: when an SAE is upgraded into an executable certificate of boundary behaviour on a fixed protocol family, the hidden-state dictionary it ships is much larger than the boundary-canonical complexity of the protocol itself. The framework therefore predicts, and the data confirm, that a global tree or distillation surrogate will dominate any SAE on the boundary-certification axis whenever $\mathcal{Q}_n$ is small relative to $d_h \cdot n_{\mathrm{atoms}}$.

Table 6: SAE flagship: ledger of an SAE-based explanation of three small LMs on the 512-prompt protocol. Visible = dictionary + sparse codes + linear probe. Adapter = sparse-coding algorithm spec. Correction = boundary mismatch patching only; no separate hidden-state residual term is charged because the target notion of equivalence is boundary behavior, not exact hidden-state reconstruction.

Teacher	visible (kbit)	adapter (kbit)	correction (kbit)	full (kbit)	vis. fid.
distilgpt2	1 770	0.256	1 574	3 344	†
gpt2	1 770	0.256	1 574	3 344	†
pythia-410m	2 302	0.256	2 097	4 400	1.000

† Linear probe degenerate on this prompt set; correction column already pays the full mismatch.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the theoretical claim (exact certified explanations compile into simulators and are lower-bounded by boundary-canonical complexity) and the empirical claim (apparent compression often disappears under full-cost accounting). These match Sections 4, 5, and 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated Limitations section discussing the finite-horizon scope, the use of code-length proxies, the status of certificates in neural audits, and the gap between executable certification and human usefulness; see Section 7.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual papers cannot address all potential limitations.

3. **Theory assumptions and proofs**
   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?
   Answer: [Yes]
   Justification: The main text states the assumptions and theorem statements, while complete proofs and formal definitions are given in Appendix A and Appendix B. Additional variants and worked examples appear in the supplementary appendices.
   Guidelines:
   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the main paper should include a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented with formal proofs provided in appendix.
   - Theorems and lemmas that the proof relies upon should be properly referenced.
4. **Experimental result reproducibility**
   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?
   Answer: [Yes]
   Justification: The paper specifies the ledger schema, experiment families, protocol definitions, seeds, audited domains, and compute budget in Section 6 and Appendix D, together with codebook and reporting rules in Appendix E.
   Guidelines:
   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, the paper should make them reproducible to the extent that it affects the main claims and/or conclusions of the paper. It should not matter if the authors prefer not to include an Open Access link to a code repository, if there is sufficient information for someone to reproduce the experiment. This may depend on how standard the data and software are used, and how easy those are to obtain.
   - Sufficient information should be provided to match the experimental setup used to generate the main results, with a clear enough description to support replicability.
   - Important parameters for the experimental setting should be reported, e.g. model selection criteria, margin of error or sample sizes depending on the application, and the rationale for the dataset choice.
   - The authors should make clear whether the results are exact on exhaustive finite domains, measured on sampled audit sets, or approximate under a stated distribution.
5. **Open access to data and code**
   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?
   Answer: [Yes]
   Justification: The submission includes the accompanying code and paper-source folders prepared for release, and the paper states that the code is provided with the submission; see Section 8 and Appendix D.
   Guidelines:
   - The answer [N/A] means that paper does not include experiments requiring code and data.
   - For papers with code, the answer should be [Yes] if the code is uploaded with the paper,

even if anonymity is preserved by not providing a public URL until after review.
- The answer should be [No] if the authors do not plan to release code or data.

6. **Experimental setting/details**
   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?
   Answer: [Yes]
   Justification: The experimental protocol appendix reports the audited domains, task construction, model families, training settings, seeds, split sizes, and package families for the Boolean, MLP, DFA, CNN, and LLM audits; see Appendix D.
   Guidelines:
   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be described precisely to permit understanding and reproduction.
   - Important hyperparameters and optimization settings should be provided.
   - Data splits and evaluation protocols should be clearly defined.
7. **Experiment statistical significance**
   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?
   Answer: [No]
   Justification: Some experiments are exhaustive over finite domains and therefore do not require statistical uncertainty estimates, while seeded experiments report repeated runs in the saved metrics. However, the main paper does not consistently present error bars or formal significance analyses for all non-exhaustive experiments.
   Guidelines:
   - The answer [N/A] means that the paper does not include experiments.
   - Experiments should provide error bars, confidence intervals, standard deviations, or another appropriate uncertainty summary whenever randomness or sampling meaningfully affects the results.
   - If exact exhaustive evaluation is used, that should be made clear.
8. **Experiments compute resources**
   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?
   Answer: [Yes]
   Justification: Appendix D includes a reproduction-budget table with per-experiment row counts, hardware, wall-clock times, and memory-related notes, plus additional hardware-envelope discussion.
   Guidelines:
   - The answer [N/A] means that the paper does not include experiments.
   - The paper should disclose sufficient compute information to let reviewers understand practical reproducibility.
   - Worker type, memory, and typical runtime should be provided when relevant.
9. **Code of ethics**
   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?
   Answer: [Yes]
   Justification: The work is theoretical and empirical on synthetic tasks plus publicly available pretrained models, does not involve deception of human participants, and discusses scope and limitations. We are not aware of any aspect that conflicts with the NeurIPS Code of Ethics.
   Guidelines:
   - The answer [No] should be used if there are ethical concerns or known departures from the code.
   - Authors should briefly justify the answer.
10. **Broader impacts**
    Question: Does the paper discuss both potential positive societal impacts and negative societal

impacts of the work performed?

Answer: [No]

Justification: The paper does not include a dedicated broader-impact discussion covering both positive and negative societal impacts. Its discussion is focused on technical scope, limitations, and reproducibility.

Guidelines:

- The answer [N/A] means that there are no reasonably foreseeable societal consequences of the work.
- If the work could plausibly affect society, the paper should discuss both positive and negative impacts.

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The submission does not release new pretrained language models, datasets, or generative systems; it studies synthetic tasks and audits existing third-party pretrained models. The released asset is code for the accounting and evaluation pipeline rather than a new high-risk model release.

Guidelines:

- The answer [N/A] is appropriate if the paper does not release assets with elevated misuse risk.
- If high-risk assets are released, safeguards should be described.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No]

Justification: The paper credits the sources of the main external assets through citations and model names, but it does not provide a consolidated in-paper accounting of the licenses and terms of use for every pretrained model and software dependency. This should be added in a later revision if required.

Guidelines:

- The paper should credit existing assets and respect their licenses and terms of use.
- If the licensing status is not fully documented in the paper, the answer should be [No].

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The new assets are the released code, saved metrics, and paper-source package. They are accompanied by README files and a protocol appendix describing the experiment schema, package fields, and reproduction setup; see Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not introduce new assets.
- New assets should be documented and accompanied by usage information.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, or any research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing or research with human subjects.

- If human-subject experiments exist, the paper should include the relevant instructions and compensation details.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human subjects, so IRB approval and participant-risk reporting are not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve human subjects.
    - If the work involves human subjects, the relevant approvals and risk disclosures should be documented.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Pretrained language models are an explicit experimental domain in Experiment 6, with the protocol family, audited models, and package construction described in Section 6 and Appendix D. They are part of the empirical evaluation rather than hidden tooling.

    Guidelines:

    - The answer [N/A] is appropriate if LLMs were used only for writing, editing, or formatting and are not part of the scientific contribution.
    - If LLMs are part of the methods or evaluation, the paper should describe that usage.