Nick Clark · Published July 2, 2026 · PDF

How to Prove an AI Model Was Trained Only on Licensed Data

Teams that fine-tune or pretrain models increasingly need to answer a hard question from licensors, regulators, and their own legal counsel: can you prove what your model learned from, and under what rights? This guide walks through an architecture that produces that proof at training time instead of reconstructing it afterward. It describes an approach disclosed in United States Patent Application 19/647,395, the Training Governance inventive step, not a shipping library you can install.

What You Are Building

You are building a training pipeline that can produce a defensible, auditable answer to two questions: which content entered the model, and under what rights was each piece admitted. The searcher who lands here usually has a concrete version of that problem. A licensor asks whether their catalog was used. A regulator asks for evidence that restricted content was not deeply integrated. Counsel asks whether an expired license still lives inside the weights. A conventional training run cannot answer any of these with confidence, because it keeps no record of what it learned from.

The goal is a pipeline where admission is a gated decision, every admitted example carries a rights profile, and every decision lands in a tamper-evident log you can query after training. This guide describes the architecture disclosed in United States Patent

Application 19/647,395. It is a design you implement yourself, not a package you download.

Why the Obvious Approaches Fall Short

The usual answer is to keep a manifest. You record the datasets you sampled, store the license documents alongside them, and treat that spreadsheet as your proof. This is genuinely useful for coarse questions, but it has a structural gap: the manifest describes the corpus, not the training run. It cannot tell you whether a specific example was actually admitted, whether it was skipped, or how much it influenced the model. When a licensor asks about one work rather than one dataset, a corpus-level manifest cannot resolve the question.

The second obvious approach is post-hoc unlearning: train freely, then remove offending content later if a dispute arises. The filed disclosure is direct about why this is weak. The influence of any single training example on a deep network is diffused across millions or billions of parameters through the non-linear dynamics of gradient-based optimization, which makes it impossible to precisely identify and reverse the parameter changes attributable to that example. Unlearning is therefore approximate, stochastic, and irreversible. You cannot prove you removed what you cannot prove you isolated.

Neither approach treats the training loop itself as the place where governance happens. In conventional systems the loop is an ungoverned optimization process: data is sampled, gradients are computed, the optimizer updates, and every example contributes to every layer with equal structural authority. There is no admission decision to point to, and so no proof to produce.

The Architecture

The disclosed approach reconceives the training loop as a governed execution environment, where each training iteration is treated as a proposed change to the model's knowledge that must be evaluated for admissibility before it is committed. Four elements make that possible.

Semantic metadata on every example. Before an example can be admitted, it must carry metadata sufficient to evaluate it. The disclosure specifies at minimum: an entropy band classification (semantic complexity and information density), a slope position (the content's place in a trust hierarchy), a content provenance record (source, acquisition pathway, and chain of custody), and a policy scope (the licensing terms, usage restrictions, temporal validity bounds, and exclusion mandates that apply). An example that arrives as raw content with no accompanying metadata cannot be evaluated and is therefore inadmissible by default. This turns the corpus from an undifferentiated mass of data into a governed collection of annotated objects.

An admission gate at the loop boundary. A substrate sits at the boundary between the forward-pass loss computation and the backward-pass gradient application. Gradients are computed as usual, but before they are applied the substrate evaluates the example against its policy scope and provenance. A key consequence is that non-training is a valid result, not an error: if an example fails validation, the substrate may reject it, producing an iteration in which no parameters update. That rejection is recorded. The substrate does not change the mathematics of gradient computation or the optimizer; it governs which gradient signals reach the model and with what magnitude.

Rights-bound depth profiles. The gate is not merely admit or reject. The same policy objects that govern content access elsewhere on the platform are consulted during training to assign a depth profile: a per-block weight vector controlling how far into the model an example's gradient is permitted to flow. The disclosure ties this directly to rights. Freely licensed content is encoded deeply and durably. Content

admitted under a time-limited license receives a suppressed profile that confines its influence to shallow layers, so it stays structurally separable and can be de-emphasized later without model-wide surgery. Content from a governed exclusion corpus receives a zero-weight profile that lets no gradient reach any layer. When multiple policies apply, the most restrictive one wins, and that resolution is deterministic and recorded. The point of proof is structural: the model's knowledge structure comes to reflect the rights under which each piece was admitted.

An append-only training provenance log. For each batch or example, the substrate records the entropy band, slope position, the depth-aggregation profile that was applied, the actual per-block contribution weights, the policy object that authorized admission, the content provenance record, and the admissibility determination with its reason. The log is chronologically ordered and append-only: entries are timestamped and sequentially numbered, so they cannot be modified, deleted, or reordered without producing detectable inconsistencies. The disclosure notes the log may be periodically sealed using cryptographic sealing infrastructure to produce tamper-evident checkpoints for third-party verification.

This log is what makes the proof concrete. When a content owner asks whether their content was used, the log gives a definitive answer: either the content is present with its provenance record, depth profile, and contribution weights available, or it is absent and the log confirms the absence. A reverse query starts from an observed model behavior and traces back to the training content whose depth profiles encompassed the layer blocks active during that behavior. The disclosure is careful here: the reverse query does not definitively attribute behavior to specific content, because the non-linear dynamics of optimization preclude exact attribution, but it produces a bounded attribution set substantially narrower than the full corpus.

How to Approach the Build

Step 1: Make metadata a hard admission precondition. Enrich every example before training with its entropy band, slope position, provenance record, and policy scope. Enforce the default rule from the disclosure: no metadata means inadmissible. Do this in the data loader so the gate downstream can rely on it.

Step 2: Represent rights as policy objects, not free text. A license needs machine-resolvable fields: what integration depth it permits, an expiry, and a revocation state. The same object should be usable wherever content access is governed, so training is not implementing its own separate rights logic.

Step 3: Interpose a gate between loss and gradient application. Illustrative interface sketch, faithful to the disclosed boundary and not a working library:

```
# Illustrative only. You implement this.
def admit(example, policy_store):
    if example.metadata is None:
        return Reject("no semantic metadata")
    policies = policy_store.resolve(example.policy_scope)  # most-restrictiv
    if policies.excluded:
        return Admit(depth_profile=zero_weight())          # gradient reache
    if policies.expired or policies.revoked:
        return Admit(depth_profile=zero_weight())
    if policies.time_limited:
        return Admit(depth_profile=suppressed())           # shallow layers
    return Admit(depth_profile=full_depth())               # freely licensed
```

Step 4: Enforce the depth profile on the backward pass. The disclosure describes three interchangeable techniques for applying a per-block weight vector to gradients: gated residual connections, attention-based depth selection, and architecture-agnostic layer-specific scaling factors. The scaling-factor technique is the

simplest to retrofit: scale each example's gradient at each block boundary before it accumulates into the block's gradient buffer. Apply this on the backward pass only, so forward-pass inference behavior is unchanged, and note it works with standard optimizers because it alters the gradient the optimizer receives, not the update rule.

Step 5: Write the provenance log as you go, append-only. Emit one structured entry per example or batch with the fields listed above. Timestamp and sequentially number every entry. Periodically seal checkpoints so a third party can verify the log was not rewritten after the fact.

Step 6: Build the two query paths. A forward query from a piece of content to the blocks it influenced, and a reverse query from a behavior to the bounded set of content that was structurally permitted to influence the active blocks. These are what you hand an auditor or a licensor.

Step 7 (optional): Add memorization classification. When inference output resembles a known training artifact, a reverse query against the log lets you classify the similarity as shallow (governed, confined to shallow layers), deep (integrated into deep layers, which may be compliant or may flag a governance failure), or absent (no record of the content). This connects a downstream complaint back to a training-time decision.

What This Does Not Give You

This is an architecture, not a drop-in library. There is no package to install and nothing that "just works" out of the box; you implement the gate, the depth-profile mechanism, the log, and the query paths in your own stack. The approach is disclosed in a patent filing. It is not benchmarked, productized, or presented here with performance numbers, and the disclosure does not supply any.

Be honest about the ceiling on attribution. The disclosure states plainly that exact attribution of a behavior to a specific example is precluded by the non-linear dynamics of optimization; the reverse query yields a bounded set, not a single culprit. The proof this architecture produces is a proof of governed admission and recorded influence, which is a different and stronger claim than proof of exact causal attribution inside the weights.

The approach also depends on its inputs. If your metadata is wrong, your policy objects mis-encode a license, or content enters through a path that bypasses the gate, the log faithfully records a governed process built on bad premises. Content whose origin cannot be structurally verified is flagged provenance-incomplete and, by policy, may be restricted to shallow layers; that is a mitigation, not a substitute for clean provenance at the source.

Disclosure Scope

The architecture described in this guide is disclosed in United States Patent Application 19/647,395. This guide is educational: it explains an approach a developer can study and implement, and it is not a warranty, a benchmark, a compliance certification, or an offer of software. Nothing here is a shipping product or a guarantee of any legal or regulatory outcome. Implementing the architecture is the reader's responsibility, and any claims made to licensors, regulators, or courts remain the implementer's to substantiate.

Training Governance (/training-governance) All 40 steps → (/inventive-steps)

Govern what the model learns, at what depth, with what provenance.

Chapter 11 (/patents/19-647395/chapters/training-governance)

PRIMARY TECHNICAL DISCLOSURE

- Depth-Selective Training Governance for Machine Learning Systems (/articles/depth-selective-training-governance-for-machine-learning-systems)

SECONDARY TECHNICAL

- Training Examples as Proposed Semantic Mutations in Governed Training (/articles/training-governance/mutation-proposals)
- Entropy-Band-Indexed Training Depth Profiles (/articles/training-governance/entropy-depth-profiles)
- Depth-Selective Gradient Routing for Governed Training (/articles/training-governance/gradient-routing)
- Training-Level Memorization Detection (/articles/training-governance/memorization-detection)
- Differential Privacy Through Depth-Selective Routing (/articles/training-governance/differential-privacy)
- Governed Fine-Tuning With Verifiable Provenance (/articles/training-governance/fine-tuning-provenance)
- The Training Loop as a Governed Execution Environment (/articles/training-governance/governed-training-loop)
- Policy-Governed Knowledge Retention and Suppression (/articles/training-governance/knowledge-retention)
- Provenance-Traceable Training Dynamics (/articles/training-governance/provenance-tracing)
- Curriculum-Integrated Depth Scheduling (/articles/training-governance/curriculum-depth)
- Affect-Modulated Training Depth (/articles/training-governance/affect-modulated-depth)
- Training-Inference Governance Integration (/articles/training-governance/training-inference-integration)
- Training Governance for Human-Relatable Agents (/articles/training-governance/human-relatable-training)
- Governed Training with Depth-Selective Gradient Aggregation (/articles/training-governance/edge-fleet-training)

APPLICATIONS · GENERAL

- Rights-Compliant Model Training Through Depth-Selective Gradient Routing (/articles/training-governance/rights-compliant-training)
- Regulated Industry Model Governance: Verifiable Training Provenance for AI Compliance (/articles/training-governance/regulated-model-governance)

- Training Governance for Medical AI: Auditable, Depth-Controlled Clinical Model Training (/articles/training-governance/medical-ai-training)
- Training Governance for Legal AI: Encoding Precedent Hierarchy Into the Model (/articles/training-governance/legal-ai-training)
- Training Governance for Financial AI: Auditable Model Risk Management Under SR 11-7 (/articles/training-governance/financial-model-training)
- Training Governance for Defense AI (/articles/training-governance/defense-ai-training)
- How to Train an Educational AI Tutor That Learns Pedagogy Deeply and Misconceptions Shallowly (/articles/training-governance/educational-model-training)
- Copyright-Compliant Training for Creative and Generative AI Models (/articles/training-governance/creative-ai-training)
- Training-Data Provenance for Regulated Autonomy: UNECE, FDA, and EU AI Act Compliance (/articles/training-governance/regulated-autonomy-training)
- Tamper-Evident Fleet Training Records for Maritime and Agricultural Operations Without Cellular Connectivity (/articles/training-governance/maritime-fleet-training)
- 21 CFR Part 11 Compliance for AI Model Training: Audit Trails, Electronic Signatures, and Provenance (/articles/training-governance/cfr-21-part-11)

APPLICATIONS · SPECIFIC

- OpenAI Training vs Governed Depth-Selective Training (/articles/training-governance/openai-training)
- Anthropic Constitutional Training vs Governed Training: What Depth-Selective Provenance Adds (/articles/training-governance/anthropic-training)
- Stable Diffusion Training vs Governed Provenance: The Missing Layer in Stability AI's Pipeline (/articles/training-governance/stability-ai)
- Midjourney vs Governed Training: Depth-Selective Provenance for Generative Art (/articles/training-governance/midjourney)
- Scale AI Alternative: Governed Learning Beyond Data Labeling (/articles/training-governance/scale-ai)
- Labelbox Alternative for Governed Training: Annotation Workflows vs Learning Dynamics (/articles/training-governance/labelbox)
- Snorkel AI Programs Labels but Does Not Govern Gradient Depth (/articles/training-governance/snorkel-ai)
- Weights & Biases Alternative for Governed Training: Tracking vs Depth-Selective Governance (/articles/training-governance/weights-biases)
- Determined AI Alternative: Governed Training Beyond Compute Orchestration (/articles/training-governance/determined-ai)

- MosaicML Optimizes Training Efficiency, Not Learning Governance (/articles/training-governance/mosaic-ml)
- Tesla Shadow Mode vs Governed Fleet Training (/articles/training-governance/tesla-shadow-mode)
- Symbotic Warehouse Automation vs Governed Training Provenance (/articles/training-governance/symbotic-warehouse)
- Governed Alternative to Google Gemini and Vertex AI Tuning (/articles/training-governance/google-gemini-tuning)
- OpenAI Fine-Tuning and RFT vs Governed, Depth-Selective Training (/articles/training-governance/openai-fine-tuning-rft)
- PathAI Alternative: Governed Digital-Pathology Training (/articles/training-governance/pathai-pathology)
- Tempus AI vs Governed Medical-AI Training: Training-Step Provenance (/articles/training-governance/tempus-ai-medical)

Training Governance overview → (/training-governance)