Nick Clark · Published April 25, 2026 · PDF

How to Build an Auditable, Rights-Compliant AI Training Pipeline

If you train models on licensed, time-limited, or rights-restricted data, you eventually face a hard question: what actually entered the model, and how deeply? This guide describes an architecture that governs data admission at training time and keeps an auditable record of every integration decision, so downstream behavior traces back to governed inputs. It describes an architecture disclosed in United States Patent Application 19/647,395, not a shipping library, and it centers on the Training Governance inventive step.

What You Are Building

You are building a training pipeline that can answer, with evidence, three questions that most pipelines cannot answer at all: which data was permitted to influence the model, how deeply each piece of data was integrated, and under what policy each decision was made. The goal is a pipeline where a licensing team, a regulator, or a content owner can ask "was my content trained on, and to what extent?" and receive a definitive, recorded answer rather than a shrug.

This is the problem faced by anyone training on content that carries obligations: time-limited licenses, per-creator terms, jurisdictional restrictions, or exclusion mandates. Once data has flowed through a conventional training loop, its influence is diffused

across billions of parameters and effectively unrecoverable. The architecture described here moves governance to the moment of integration, so that "governed" is a structural property of the trained model rather than a promise about the corpus.

This guide teaches the architecture disclosed in United States Patent Application 19/647,395. It is a design you implement yourself, not a package you install. Nothing here is a benchmarked product or a runnable SDK.

Why the Obvious Approaches Fall Short

The conventional way to handle rights-restricted training data is dataset-level curation: filter, weight, or exclude examples before they enter the training loop. This is real and useful, but it operates entirely upstream of optimization. Once an example passes the filter, it contributes to every layer's parameter updates with equal structural authority. The pipeline has no mechanism to control how deeply a given example integrates, and no mechanism to record which examples influenced which capabilities. Curation decides membership in the corpus; it does not govern integration into the model.

The obvious remedy when something goes wrong is post-hoc unlearning: after training, estimate a specific example's influence on the parameters and apply corrective updates to remove it. This is a legitimate research area, but it is inherently approximate. The influence of any single example is spread across millions or billions of parameters through the non-linear dynamics of gradient descent, so the correction is an estimate, not a reversal. If a license expires and you need to demonstrate that the content is no longer deeply encoded, approximate attenuation is a weak evidentiary position.

The third common approach is to bolt governance onto inference: filter outputs, add system-prompt rules, or run a similarity check against known artifacts. That governs what the model says, not what it learned. It cannot tell you where restricted content lives inside the weights, and it does nothing for the auditor who wants to see that a depth restriction was actually applied during training.

The structural gap in all three: governance sits before or after the training loop, never inside it. The training loop remains an ungoverned optimization process.

The Architecture

The disclosed approach reconceives the training loop as a governed execution environment. Each training iteration is treated as a proposed change to the model's knowledge state that must be evaluated for admissibility before it is committed, exactly as an inference-time transition or an agent state change would be. The same policy objects, the same provenance infrastructure, and the same admissibility logic that govern behavior elsewhere on the platform are extended into training.

A substrate at the gradient boundary. The spec positions a semantic execution substrate at the boundary between the forward-pass loss computation and the backward-pass gradient application. Gradients are computed as in conventional training, but before the gradient signal reaches the model's parameters, the substrate evaluates the training example and decides how that gradient is gated, scaled, or routed across the model's depth. Critically, the substrate does not alter the mathematics of gradient computation or the optimizer's update rule. It governs which gradient signals reach which layers, and with what magnitude, based on the semantic properties of the content that produced them.

Every example carries metadata, or it is inadmissible. Raw content with no accompanying metadata cannot be evaluated and is inadmissible by default. The spec specifies that each example carries, at minimum: an entropy band classification (semantic complexity and information density, derived from the platform's entropy extraction pipeline); a slope position within the platform's trust hierarchy; a content provenance record identifying source, acquisition pathway, and chain of custody; and a policy scope naming the licensing terms, usage restrictions, temporal validity, and exclusion mandates that apply. The corpus becomes a collection of governed, annotated objects rather than an undifferentiated mass.

Admission is graded, not binary. The substrate can admit, reject, or admit-with-modification. A rejected example produces a training iteration in which parameters are not updated, and that non-training event is recorded as a governed result rather than an error. A modified admission attaches a depth profile: a per-block contribution weight vector that specifies, for each layer block, how much of that example's gradient is permitted to flow. A weight of zero blocks the gradient at that block entirely; a weight between zero and one attenuates it; unit weight passes it through.

Depth-selective aggregation is the enforcement mechanism. The depth profile is applied to the gradient signal during the backward pass. The spec describes three interchangeable techniques for this, chosen to fit the architecture: gated residual connections (a per-layer gating coefficient on the residual pathway during backprop), attention-based depth selection (scaling the gradient reaching attention and value projections in transformers), and a model-agnostic layer-specific scaling factor (a scalar multiplier applied to the gradient at each layer boundary before accumulation). All three modulate the backward pass only; the forward pass runs with standard connections, so inference behavior is unaffected by the mechanism. Per-example scaling happens after per-example gradient computation and before batch accumulation, and the optimizer (SGD, Adam, AdamW) then treats the modulated buffer as an ordinary gradient buffer.

Policy drives depth. This is where rights-compliance becomes structural. The spec describes mapping content classes to depth profiles through the same policy objects used elsewhere: content under a time-limited license is trained with a suppressed depth profile (deep-layer weights at or near zero), confining its influence to shallow, separable layers so that later de-emphasis does not require full retraining. Content in a governed exclusion corpus gets a zero-weight profile at every layer, so it influences no parameters at all. When multiple policies apply to one example, the substrate resolves them by applying the most restrictive, and records which policy governed the outcome. The spec

frames this as structural prevention rather than post-hoc unlearning: there is nothing to unlearn if the content was never deeply learned, and a zero weight at a block is exact, not an estimate.

The training provenance log makes it auditable. For each batch or example, the substrate writes a structured entry: entropy band, slope position, the depth-aggregation profile applied, the actual per-block contribution weights that reached each block, the governing policy object, the content provenance record, and the admissibility determination with its reason. The log is chronologically ordered and append-only, timestamped and sequentially numbered by epoch, iteration, and batch, so tampering produces detectable inconsistencies. The spec notes the log may be periodically sealed using cryptographic sealing infrastructure disclosed in a cross-referenced governance filing, producing tamper-evident checkpoints for third-party verification.

Queries turn the log into answers. A forward query starts from an example or content class and traces which blocks it was permitted to influence and by how much. A reverse query starts from an observed behavior and traces back to the training content whose depth profiles encompassed the active blocks. The spec is explicit and honest about the limit here: a reverse query does not definitively attribute behavior to specific content, because gradient-based optimization precludes exact attribution. What it yields is a bounded attribution set that is substantially narrower than the full corpus. For a content owner's "was my content used" inquiry, the log gives a definitive yes-with-records or no-with-confirmation.

How to Approach the Build

The following order reflects the architectural dependencies in the spec. Each step is something you design and implement.

1. Build the semantic enrichment stage first. Nothing downstream works without per-example metadata. Stand up the pipeline that, for every candidate example, computes an entropy band classification, resolves a slope position, assembles a content provenance record (source, acquisition pathway, chain of custody), and attaches a policy scope. The spec derives entropy from the information-theoretic divergence of the example's semantic embedding relative to the model's current representational state (it names KL or Jensen-Shannon divergence). Treat "no metadata" as inadmissible by default from day one, so ungoverned data cannot slip through later.

2. Model policy objects as first-class, shared entities. The whole point is that training consults the same policy objects that govern the rest of your system. Design a policy object that can express licensing terms, temporal validity, usage restrictions, and exclusion status, and a resolution routine that, given multiple applicable policies, returns the most restrictive result deterministically. An illustrative interface sketch, faithful to the spec and not a working implementation:

```
# Illustrative only. You implement the real logic.
resolve_depth_profile(example.metadata, applicable_policies) -> DepthProfile
#   returns per-block weights; time-limited -> suppressed (deep blocks ~0)
#   exclusion corpus / expired / revoked -> zero-weight (all blocks 0)
#   picks the most restrictive when policies conflict; logs which policy won
```

3. Choose your depth-selective mechanism to match the model. For a transformer, attention-based depth selection or block-level scaling factors fit naturally. For an arbitrary architecture, start with the model-agnostic layer-specific scaling factor, since it only requires intercepting and scaling the gradient at each layer boundary during the backward pass. Work at block-level granularity, not per-layer: the spec notes per-layer profiles are unwieldy in deep networks, so group layers into architecturally meaningful blocks and apply one weight per block.

4. Insert the substrate at the gradient boundary. Wire the admission and depth-modulation step between per-example gradient computation and batch accumulation. Verify the forward pass is untouched. A useful invariant to assert in tests: with all policies permissive and all weights at one, the modulated gradient buffer must be numerically identical to the unmodulated one, so governance adds no behavior when it should be a no-op.

5. Write the provenance log as append-only from the start. Record every determination, including rejections and zero-weight exclusions, with the full structured entry the spec lists. Sequential numbering plus timestamps per epoch, iteration, and batch is what makes tampering detectable; retrofitting this later is painful, so build it in now. If you have sealing infrastructure, checkpoint-seal the log periodically.

6. Implement forward and reverse queries against the log. These are what convert your log from a write-only artifact into a compliance tool. Build the content-owner inquiry path (is this content present, and with what profile and weights) and the reverse-query path (given a behavior, return the bounded set of content that was structurally permitted to influence the active blocks). State the reverse query's bounded-set semantics plainly in your own tooling so no one mistakes it for exact attribution.

7. Optionally add the two-dimensional controls. The spec pairs a curriculum engine (temporal: when each entropy band is presented) with depth profiles (spatial: where content is integrated), and describes a profile adaptation engine that adjusts profiles at checkpoints based on the model's measured layer-wise entropy. These are enhancements; the auditability and rights-compliance core does not depend on them.

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." You implement the enrichment pipeline, the policy objects, the gradient-boundary substrate, the depth-selective mechanism, and the provenance log yourself, and you make them fit your model and your training stack.

The approach is disclosed in a patent filing. It is not a benchmarked or production-proven system, and this guide reports no performance numbers, convergence claims, or accuracy results, because the spec asserts none for you to rely on. Whether depth-selective routing helps or hurts your model's quality on your data is something you must measure.

Be honest about attribution. The spec is careful that reverse queries yield a bounded attribution set, not a proof that a specific example caused a specific behavior; exact attribution is precluded by the nature of gradient-based optimization. The provenance log gives you definitive presence-and-depth records, which is a strong compliance position, but it is not a causal microscope on model behavior.

Finally, the guarantees are structural, not magical. A zero-weight block is exact only if your mechanism is implemented correctly and every example is genuinely enriched and evaluated. The default-inadmissible rule for unmetadata'd content is what keeps that assumption honest; if you weaken it, you weaken everything downstream.

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 design and build, and it is not a warranty, a guarantee of results, or an offer of software. References to general technologies and methods (for example, KL or Jensen-Shannon divergence, residual connections, attention mechanisms, and standard optimizers such

as SGD, Adam, and AdamW) are provided for context and are described as they are used in the disclosed architecture. Implementation choices, performance, and compliance outcomes are the responsibility of the party building the pipeline.

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)