When to Adopt Model Updates

Aspen Kennedy Hopkins* **Isabella Struckman** **Hedi Driss** **Aleksander Madry**
CSAIL, Massachusetts Institute of Technology
Cambridge, MA, USA

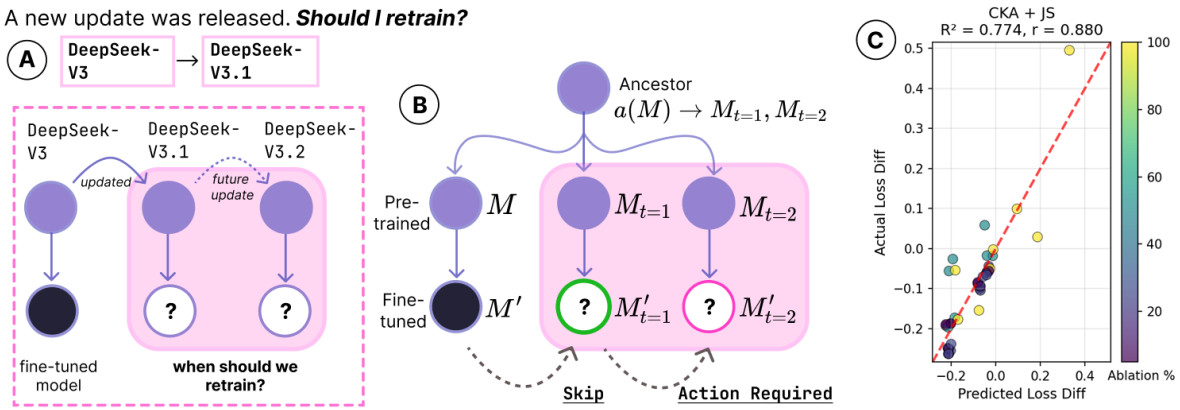

Figure 1: (A) shows the current state of update adoption, where developers rely on limited information to make costly, ad-hoc retraining decisions. (B) depicts model lineages and the information gained using our proposed framework. By characterizing structural relationships, our method provides actionable signals indicating which updates require propagation and which can be safely skipped. (C) Shows the results of a linear regression model predicting output divergence between M_1 and M'_1 ($r = 0.880$) for Qwen2.5-3B-parameter LLMs.

Abstract

As AI development increasingly involves models adapted to downstream settings, a new algorithmic challenge for developers has surfaced: deciding when to adopt model updates. Currently, the only reliable approach is to retrain on new upstream models and then evaluate extensively, which can be prohibitively expensive. We introduce a framework for assessing when to propagate newly released model versions to downstream applications **that does not require access to upstream data or retraining *a priori***. Our framework incorporates geometric similarity and information-theoretic sufficiency to determine when an upstream update substantially shifts the representational basis recruited for a downstream task. This enables targeted update adoption, and supports emerging norms for coordinating AI supply chain infrastructure.

1 Introduction

Model lifespans have become increasingly brief. After a model is released, it will at some point be replaced, no longer meeting some criteria of its provider, with an updated version. Contemporary

*Correspondence to: dataspen@mit.edu.

Preprint.

AI systems then build upon these upstream[2] pre-trained models via fine-tuning or other adaptation methods to serve specialized tasks. This layered development—the result of increasingly complex *AI supply chains*—has inspired a fundamental shift in AI system ownership, with numerous dependencies now crossing organizational boundaries [Bommasani et al., 2023, Horwitz et al., 2025b, Hopkins et al., 2025b].

Despite a growing dependence on evolving upstream models, there are currently few tools for managing or coordinating changes in these dependencies. Models are updated for numerous reasons: to improve the quality of the model, to patch security failures, to be in compliance with regulation or legal requirements, or to align with new research or business interests. These upstream modifications often yield uneven downstream effects. Updated models may perform well on upstream benchmarks, for example, but regress in particular downstream use-cases, frustrate users with out-of-date expectations of model behavior [Bansal et al., 2019], or present no practical improvement post-adoption.

Model updates, and more broadly the accumulating overhead of AI dependencies, represent an emerging algorithmic challenge. *Understanding the differential impacts of model updates represents a critical gap in our ability to coordinate AI systems at scale.*

1.1 Contributions

In this work, we focus on a specific coordination challenge in the AI ecosystem: *deciding what to do when a model is updated*. Borrowing from prior work on model families [Horwitz et al., 2025a], we first offer a definition of model updates that differentiates between temporal succession and parameter inheritance (see Figure 1AB). We then empirically illustrate that uniform adoption strategies may not be necessary across dependencies. Specifically, we show that downstream applications can be disrupted unevenly by upstream modifications to models. While intuitive, these results suggest that update propagation may be tailored to downstream domains.

We then consider what coordination signals might circumvent costly retraining or re-evaluation. Our response is inspired by a growing body of work that explores how models' internal representations (e.g., weights, activations) can be used to understand model behavior [Olah et al., 2018, Han et al., 2026, Bereska and Gavves, 2024]. Just as individual models recruit distinct representational subspaces for different tasks, so too might we expect downstream applications to depend on identifiable subsets of upstream internal representations. We do this by treating representations as a data modality to surface how downstream model adaptation or use structurally varies *prior* to update adoption. Specifically, we map changes in internal representations to an operational criterion informing downstream developers of when an update is consequential to them.

We adopt two complementary approaches to this mapping that do not require retraining or access to upstream data *a priori*: the first compares geometric similarities between internal representations; the second adopts a distributional approach by extending work by Darrin et al. [2024] to ask whether information relevant to downstream behavior is preserved across the update. Our methods can be used when models are adopted "out-of-the-box" or when they are adapted downstream, i.e., via fine-tuning.

In our analyses, we find preliminary evidence that changes in activations recruited for a downstream task mirror downstream performance shift. These findings extend to fine-tuning scenarios, where we are able to predict performance changes prior to retraining. In other words, we are able to recover an unseen model's behavior divergence. We focus the majority of our experiments on settings where a model (e.g., ResNet-18 models, Qwen2.5-3B, or EleutherAI Deep-Ignorance 6.9B LLMs) is updated with different upstream data (e.g., from data deletions or additions). This allows us to isolate the mechanism inducing downstream perturbations. We then extend our analyses to other update scenarios, i.e., where quantization, model architecture changes, or safety interventions are imposed.

In sum, we present four contributions: (1) a formal definition of model updates; (2) empirical evidence that upstream interventions have heterogeneous effects across downstream domains; (3) a mapping from representational change to a decision rule for update propagation, calibrated via τ, a threshold based on representational uncertainty between equivalent models; and (4) two complementary methods that instantiate this rule, along with evidence of their predictive fidelity.

[2]We use the term "upstream" to refer to pre-trained models or their organizations, and "downstream" to refer to models or the organizations that later use or adapt them to specialized applications.

2 State of Update Management

Our interests lie in what happens *after* a model is discretized, i.e., fixed at a particular point in time, then "released" as an update. We are motivated by the observation that discretization is a structural feature of AI deployment. This is made explicit in regulated domains[3] and implicit in unregulated settings. A fixed artifact can be audited, certified, and, to an extent, have its behavior attributed—all critical elements of component reliability [Rushby, 1993, Parnas, 1972].

Integrating model updates into heterogeneous downstream applications, however, poses a number of challenges. We use this section to first articulate why the status quo of model update management is insufficient, then outline how models' internal representations might support future tooling in response.

2.1 Evolving Needs for AI Supply Chains

The rapid expansion of the AI ecosystem has driven interest in how AI systems are adopted, integrated, and scaled globally [Bommasani et al., 2023, Horwitz et al., 2025b]. As this ecosystem has developed, dependencies between and within organizations have grown. These information flows have been described as "AI supply chains" [Lee et al., 2023, Hopkins et al., 2025a]. When individual components in these networks, e.g., models, are modified, the question of how such changes are communicated and managed across the AI ecosystem has emerged.

Existing methods in model documentation remain the gold standard in machine learning. These practices improve transparency but face several limitations. Model cards [Mitchell et al., 2019], for example, are largely descriptive (not dynamic) and do not account for downstream applications. Versioning tools track artifacts, dependencies, and training metadata rather than representing the semantic differences between model versions. Naming convention is another form of communication. Updates may share an underlying base model but be tweaked (e.g., via reinforcement learning or changes to datasets and training regimes) to meet new criteria of their developer. Minor perturbations may retain the same name with only timestamp or version changes (e.g., `2025-01-01-v2`), while major releases may introduce a new convention indicating a "model family" (`Llama2` to `Llama3`). Updates do not cleanly fall under "major" or "minor" bins, however, and denomination tends to follow provider convention rather than technical consequence—an unreliable signal for developers.

These methods lack context on how *internal changes* within a model reshape behavior across heterogeneous applications, the effects of which may only appear in some inputs, domains, or downstream deployments. Traditional software supply chains address similar coordination problems through semantic versioning, syntactically-driven compatibility contracts that allow downstream dependencies to evaluate "breaking changes" without inspection [Preston-Werner, 2011, Decan and Mens, 2019]. [4] In AI, there is currently no equivalent type signature, and behavioral compatibility is statistical and task-conditional. Our goal in this work is to explore how we might map AI systems to a similar form of abstraction within the setting of model updates.

2.2 Adoption Strategies Vary By Setting

For the purposes of this paper, we focus on two update scenarios: where a model is used without modification, and where finetuning is used to adapt a model. The first is a simple condition of understanding when to switch to a candidate update. In generative models, this often involves autoregressive generation that is then evaluated via other models, humans, or benchmarks. Updating fine-tuned models introduces additional complexity as it entails retraining the model *before* evaluating it. Both evaluation and fine-tuning incur costs to developers.

[3] For example, the U.S. FDA's guidance on AI-enabled healthcare software requires predetermined change control plans explicitly because it is hard to certify a system that changes [U.S. Food and Drug Administration, 2024].

[4] Breaking changes are changes that do not meet specifications built into prior versions of a particular library, as used by downstream developers [Ochoa et al., 2022]. Breakages are generally demonstrated via compiler, binary, or backwards compatibility checks. Systems for API management differentiate between behavioural compatibility (which is determined at run-time) and compatibility that is verifiable *a priori*. Collectively, SemVer and other methods face their own challenges in both implementation and success [Ashkenas, 2014], thus modeling a new solution for AI requires accounting for their limitations along with mismatches in the artifact types that are compared.

Responses to updates vary by setting (e.g., open or closed, fine-tuned or used directly) and developer requirements. Organizations are heterogeneous in their motivation: some care if an update introduces new risks or distribution shifts, while others index on potential improvements or legal, compliance, or security implications. Closed model users with less decision-making power may care about undisclosed changes in model behavior. Open model users may update more frequently in pursuit of incremental gains, while a regulated counterpart might restrict changes to mitigate costs of re-evaluation or certification [U.S. Food and Drug Administration, 2024]. Fine-tuning-as-a-service (FTaaS) platforms abstract away many of the underlying processes that inform downstream decisions, escalating the complexity of update management. Meanwhile, large organizations maintaining multiple models face similar challenges internally, deciding when and how to deploy updates across interconnected applications through repeated cycles of training, evaluation, and deployment. Those that modify models on a contractual basis must decide when to incorporate updates *and* how to communicate these changes to their users, balancing transparency with intellectual property interests.

Collectively, these constraints heighten pressure for update propagation. Upstream changes may induce cascading evaluation and retraining across AI supply chains. Organizations may waste resources re-training models that already exist or that produce redundant outcomes [Horwitz et al., 2025b], fail to adopt models that might otherwise be beneficial, or not calibrate evaluations to capture new behavioral changes, yet tools to support heterogeneous decision-making *a priori* either do not yet exist or rely on external metadata disconnected from downstream needs.

2.3 Internal Representations as a Coordination Substrate

Previous work has shown that models process information through their intermediate layers, revealing structured representations that capture semantic, syntactic, and task-relevant features [Bengio et al., 2013]. Interpretability and weights-based methods build upon this, focusing on how the weights, internal computations, and representations within neural networks can be treated as data to understand the outputs we observe [Olah et al., 2018, Kahana et al., 2024]. Studying these representations, recent work by Huh et al. [2024] argued that as models scale and training practices converge, different models increasingly learn similar "Platonic" representations that result in a shared statistical model of the world.

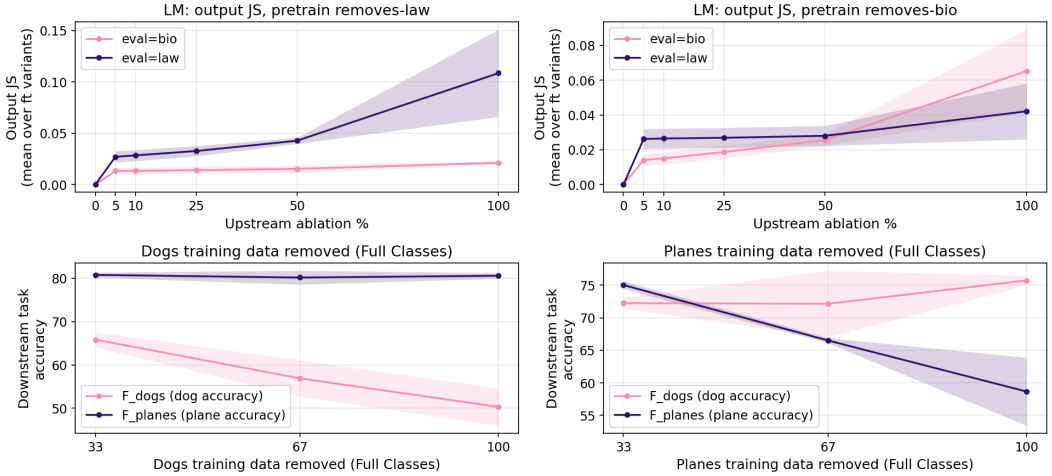

Figure 2: Upstream perturbations produce heterogeneous domain-specific consequences downstream, shown across two modalities with full fine-tuning: In Qwen models (top), JS divergence between a fine-tuned M' and upstream updates increases on the matched evaluation domain but remains comparatively flat on the unmatched domain; in vision (bottom), full fine-tuning on the matched class loses accuracy as that class's pretraining data is removed, while the cross-class fine-tune is unaffected.

Following Horwitz et al. [2025a], we observe that while models indeed share substantial representational overlap—particularly for fundamental concepts and tasks—they also maintain distinct characteristics shaped by their ancestry and training trajectories: models within the same "family"

exhibited smaller nuisance variations, or greater similarities, than those with different ancestors. In other words, Qwen2.5-0.5B-Instruct and Qwen2.5-0.5B have more in common with each other than with Llama models. This has direct implications for our work: commonalities across models, particularly within the same family, suggest that certain representations may be universal enough to study systematically, while model-specific variation can indicate between-model differences. This perspective allows us *to leverage the model itself* in diagnostic tools comparing models within the same family, e.g., for exploring how specific updates alter a model's conceptual organization and processing strategies.

In the following sections, we offer a definition of model updates that differentiates between temporal succession and parameter inheritance (Section 3). We believe this differentiation is necessary because it highlights that backwards compatibility between models is not simply a matter of tracking linear evolution. This suggests that alternative modes for flagging breaking changes are needed. We then demonstrate how the representational structure induced by parameter inheritance can be operationalized for model comparisons prior to update adoption (Section 4).

3 Model Updates

Consider a setting where an open base model is used either directly or adapted via fine-tuning for a downstream task. The base model is then updated. The key question is: *should the downstream developer adopt this update?*

Let S be the upstream dataset used to pre-train a base model $M = f(S)$. A downstream developer may use M directly, or adapt it through fine-tuning on a dataset R, which is chosen to learn a downstream distribution $\mathcal{D}$, to produce $M' = g(M, R)$. When M is used directly, $M = M'$.

In the following section, we define a model update as it relates to upstream dependencies.

3.1 Definition

Hopkins et al. [2025a] introduces a general definition of an AI supply chain as a directed graph $\mathcal{G} = (V, E)$, where each node $v \in V$ denotes an *AI component*, which is defined as either a model or a dataset, and where a directed edge $(v_j, v_i) \in E$ corresponds to an *operation* used to obtain a downstream component from its parent. This definition is general, allowing for cycles, and is inclusive of operations beyond fine-tuning. Horwitz et al. [2025a] offers a complementary definition emphasizing model inheritance: model trees, in which a directed acyclic graph maps the origin of models stemming from a base model (e.g., a foundation model) to its fine-tuned descendants, and where collections of model trees produce a "model graph." Given the setting of model updates, we adopt a formalism that more closely aligns with Horwitz et al. [2025a].

Specifically, we assume that models exist within a directed acyclic hierarchy of derivation $\mathcal{G}$ over a universe of models $\mathbb{M}$. Each model $M \in \mathbb{M}$ occupies a node in the graph and is connected to one or more direct ancestors via update transformations. We assume that there is a common ancestor within a family of updates, or model versions, and ignore updates that are not part of the family.

For any two models $M_A, M_B \in \mathbb{M}$, we define an edge $(M_A, M_B) \in E$ to indicate that M_B is an update from M_A derived via a transformation process (e.g., data deletion, fine-tuning, architecture modification, etc.) and released in sequence by the model developer. A model is annotated by the order in which it is released i.e., $M_{i=0}$ represents an initially released model, $M_{i=1}$ represents the first update, and so on. In this way, we distinguish *family membership* from *direct derivation*, as shown in Figure 1. We do this to support cases where updates are not directly derived from each other, distinguishing temporal succession from parameter inheritance. This allows *rebased* releases—updates that are ordered after previous versions but re-initialize from the anchor—to appear as siblings in the derivation graph rather than as forced children of the immediately preceding release.

Finally, let $a(M)$ denote an ancestor in our family of models $\mathbb{M}$. A model update is any transition $M_A \rightarrow M_B$ within the graph G, where there exists an ancestor $a(M_B) = a(M_A)$, i.e., that M_B shares heritable traits with M_A, and where $M_A \prec M_B$. This definition allows for variability in the magnitude of the update while preserving the continuity of lineage. While the update graph may be partially latent, we treat continuity of inheritance as a structural prior that allows us to model AI systems as nodes in an evolving ecosystem.

We next consider when an update warrants downstream action.

4 Update Propagation

Our interest lies in determining how much change an upstream update will induce in the downstream setting. Let $\{M_{i=0}, M_{i=1}, \ldots, M_{i=n}\}$ denote an ordered sequence of model updates (for brevity, we drop explicit i-index notation below while preserving this ordering). Given upstream model M, a downstream model M' (recall $M = M'$ for direct use), and a candidate update M_1, we ask whether M' will be significantly impacted by changes in M_1 so as to warrant additional action such as retraining (producing M_1') or re-evaluation. An illustration is shown in Figure 1. Increasingly, models are released *without* their training datasets, making existing methods for evaluating data influence less effective. Our goal is to understand how an update to M affects M_1 without access to the upstream pretraining dataset and without retraining and evaluating M_1'.

A naive approach to measuring the consequences of a model update is to look at performance changes between M and its update M_1 on a set of inputs relevant to the downstream developer. While such an approach is straightforward, it typically requires retraining or evaluating M_1', which we want to avoid. It also does not account for the perturbations introduced through the adaption process on the downstream model, nor is it a stringent criterion for cases like data deletions, where the goal would be to determine if M' relies on the deleted data.

Instead, we look for signal of a distribution shift between M' and M_1' conditioned on the structural relationship between the models in the update graph. We focus on the local view of a supply chain node: while the structure of $\mathcal{G}$ motivates the definition of an update, the propagation decision should be tailored to the downstream application. Given M' was adapted from M, we assume its behavior is meaningfully tied to the learned representations of M [Horwitz et al., 2025a]. If M_1 diverges too far from that representation, we may expect M' to become misaligned, suggesting that retraining is warranted.

4.1 Propagation Criteria

As discussed in Section 1, changes in recruited representations between models on the same probing set may indicate whether an upstream update perturbed the representational subspace that a downstream application depends on. This is the basis for our targeted coordination signal: rather than treating updates as uniformly consequential, the criterion is conditioned on the downstream application's representational dependencies.

Let our probe dataset be $X = (x_0, x_1, \ldots, x_n)$. We assume X reasonably covers the downstream application, which is transformed into embeddings as $h_i^0 = \phi_0(x_i), \quad h_i^0 \in \mathbb{R}^{d_0}$. Then, let the layers of a given model be $H^k = \{h_0^k, h_1^k, \ldots, h_n^k\}$, where h_i^k represents the activation of input x_i at layer k, and each layer results from applying a transformation function ϕ^k. In cases where self-attention, convolution, or an MLP are used in later layers, the transformations are applied recursively as $h_i^k = \phi^k(h_i^{k-1}), \quad h_i^k \in \mathbb{R}^{d_k}$, where ϕ^k represents the transformation function at layer k. The stacked representations at layer $k \in \mathcal{K}$ (where $\mathcal{K}$ is the set of relevant layers) is then $H^k = (h_0^k, h_1^k, \ldots, h_n^k) \in \mathbb{R}^{n \times d_k}$.

We analyze how similar (or conversely, how different) M' is to M and M_1 over the dataset X. Specifically, we define similarity metrics (distance or distributionally-based) that involve comparisons within our family of models, depending on the comparison's objective.

Let $\text{Sim}(A, B)$ be a measure of distance or divergence between two models A and B (formal definitions to follow). Then, if the objective is to determine if the update altered the upstream model's representation with regards to X, a direct comparison between M and M_1 is sufficient. Given a decision threshold τ—discussed in 4.4—we adopt the following retraining criteria:

$$\text{Flag}(M', X) = \begin{cases} 1, & \text{if Sim}(M, M_1) > \tau \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The notation above assumes that Sim is oriented as a distance or divergence. Larger values indicate larger shifts; for similarity scores where larger values indicate greater alignment, such as raw CKA, we reverse the inequality or use the corresponding distance, e.g., $D_{\mathrm{CKA}} = 1 - \mathrm{CKA}$.

There are two motivations for this evaluation. First, we might assume that changes to the base model propagate to downstream tasks without unexpected distortions—that is, fine-tuning acts approximately as a linear transformation of the upstream representation, such as when it is applied only to the final layer of a given model. Second, downstream users may deploy base models as-is but need to detect distribution shifts between model versions, signaling performance changes at deployment that *might not be apparent from the outputs on X alone*. This can occur when evaluation datasets are too narrow to expose altered inductive biases, or when decoding noise, thresholds, or label symmetries mask small but systematic internal perturbations.

When we do not assume that downstream models smoothly inherit these changes—because of fine-tuning effects, task-specific dependencies, or nonlinear interactions between the upstream and downstream training dynamics—we extend the comparison to include the downstream model M' and adopt the following:

$$\mathrm{Flag}(M', X) = \begin{cases} 1, & \text{if } \mathrm{Sim}(M, M_1) > \tau \text{ OR } \mathrm{Sim}(M_1, M') - \mathrm{Sim}(M', M) \quad > \tau \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Thus, the decision to compare to incorporate M_1 hinges on the stability of a downstream adaptation in response to upstream modifications. If M and M_1 are far apart, it suggests the update changed the upstream model significantly. This is a necessary condition for considering retraining—if nothing changed, there is no need to propagate. If M' is closer to M on X in comparison to M_1, it suggests we need to retrain.

To measure similarity, we adopt two complementary approaches. The first directly compares our family of models using distance-based metrics applied to their respective internal representations over X. The second approach takes an information-theoretic perspective, asking whether the update preserves the information that downstream behavior depends on.

4.2 Similarity By Distance

In our first approach, we define similarity with regards to a chosen distance metric such as the L2 norm, cosine similarity, or centered kernel alignment (CKA):

$$\mathrm{Sim}(A, B) = \mathrm{Agg}_{k \in K}(\mathrm{Dist}(H_A^k, H_B^k)), \tag{3}$$

where $\mathrm{Agg}(k)$ is the maximum, minimum, or average distance over set or subset of layers.

Two models can be geometrically similar yet differ in their distributional structure in ways relevant to a specific application, and vice versa. Thus, a complementary question asks whether that shift alters information that is later utilized downstream. Below, we outline an information-theoretic approach that addresses this directly.

4.3 Information Sufficiency

Recent work by Darrin et al. [2024] maps concepts from information theory to the setting of embedding models by treating embeddings as noisy channels: an input is passed through a model, the model produces a representation, and a downstream task is performed from that representation. Under this view, one representation is preferable to another when it is at least as informative for downstream decision-making. We adopt this framing for model updates by treating the activations induced by M, M_1, and M' on a shared probe set X as the channel outputs being compared.

Sufficiency formalizes when a model retains all necessary information from another model. In the update setting, M_1 is sufficient for M on X if the representations of M can be recovered from the representations of M_1 by a possibly stochastic transformation, i.e., a Markov kernel (Appendix A). When this condition holds, downstream decision rules that apply to M remain valid under M_1, and no retraining is required. Exact sufficiency is not directly checkable in our setting, so we use the relaxed notion of *information sufficiency* from Darrin et al. [2024]. If M_1 captures the downstream data X

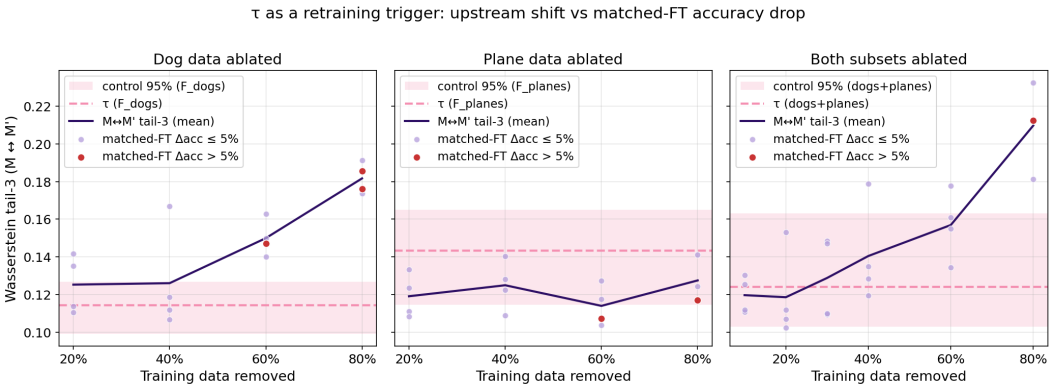

Figure 3: GMM-fitted activation distributions for ResNet models compared via Wasserstein under per-subset deletions (dog, plane, both), with per-panel control thresholds τ derived from unablated trainseeds. τ differs by panel because dog- and plane-eval sets produce distinct inter-seed variances ($\tilde{\tau}$dogs=0.114 vs $\tilde{\tau}$planes=0.143); a global threshold would be miscalibrated. Red dots mark points where matched fine-tuning accuracy drops $>$ 5pp from a clean-upstream baseline. Tightening τ yields fewer false negatives but results in increased retraining.

in a manner similar to M, then knowing $M(X)$ should allow us to predict $M_1(X)$. Information sufficiency from M to M_1 is:

$$\mathrm{IS}(M \to M_1) \triangleq \mathcal{H}(M_1(X)) - \mathcal{H}(M_1(X) \mid M(X)) \tag{4}$$

where $\mathcal{H}(M_1(X))$ is the entropy of the representations from M_1 on inputs X, and $\mathcal{H}(M_1(X) \mid M(X))$ is the conditional entropy of those representations given the representations from M. This is the standard decomposition of mutual information $I(M(X); M_1(X))$. In the three-way setting, the conditioning comparison incorporates M' to ask whether the update preserves the information that the downstream model relies on from M. High information sufficiency indicates that the updated model retains essentially the same task-relevant information about X as the original, while low sufficiency signals that information relevant to downstream behavior has been altered or lost.

Given two models, e.g., pick $A, B \in \{M, M', M_1\}$:

$$\mathrm{Sim}(A, B) = D_{\mathrm{div}}\left(P(H_A^k \mid X) \,\|\, P(H_B^k \mid X)\right), \tag{5}$$

where H^k denotes activations at a chosen layer and D_{div} is a statistical divergence, such as Wasserstein, Jensen–Shannon, or total variation distance. This allows us to ask whether an update changed the distribution of internal states on X beyond the calibrated tolerance τ.

We treat divergence-based information sufficiency as a calibrated diagnostic signal, not as a proof of exact sufficiency, under the assumption that activation distributions act as intermediate representations through which task-relevant information must pass. These similarities are then evaluated using the retraining criteria in Equations 1 and 2.

4.4 Picking τ

We treat τ as a cutoff derived from uncertainty over model representations. In the context of updates, a looser (larger) τ makes the system more tolerant to upstream changes (fewer retrains), while a stricter τ increases sensitivity at the cost of more retrains.

There are several ways to calibrate τ. The general concept is as follows. Given R replicas of downstream model M', (e.g., different seeds, LoRA-adapted replicas, or checkpoints), compute pairwise similarity $\{\mathrm{Sim}(M'_r, M'_{r'})\}$ on X and set:

$$\tau \;=\; \mathrm{Quantile}_p\Big(\{\mathrm{Sim}(M'_r, M'_{r'}) : r < r'\}\Big) \tag{6}$$

with p as chosen percentile (e.g., p=0.95). This can be interpreted as a one-sided empirical bound on representational variability induced by replicas. Intuitively, we only trigger propagation when

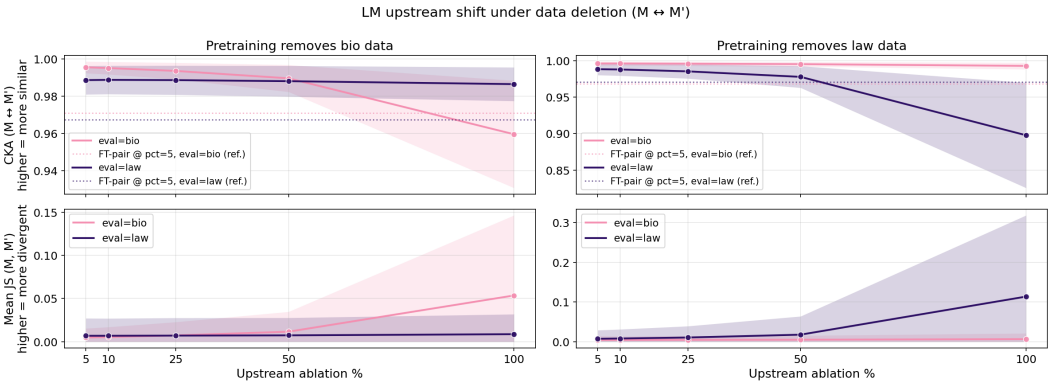

Figure 4: CKA and JS divergence between clean upstream M and ablated M' vs ablation %, faceted by intervention direction. Lines split by eval domain (bio = pink, law = purple); ablation effects concentrate on the matched eval while cross-domain similarity stays flat, mirroring the JS outputs. Bands are within-comparison spread, not seed CIs.

the observed shift exceeds typical replica variability. In our experiments, p=0.95 functions as a conservative default: it treats the upper tail of replica-to-replica variation as ignorable noise and flags only shifts outside that empirical null. Other deployments can choose p according to their tolerance for false positives versus missed updates. This calibration ensures that retraining is not triggered by differences that are indistinguishable from those observed between functionally equivalent replicas.

Replicas offer a direct way to assess natural variance. Training or accessing multiple seed variants is common in many deployment settings and provides an ideal setting for robustly calibrating τ. In practice, however, there are instances where developers do not have access to replicas. Here, applying noise to the weights, using different training checkpoints, or incorporating lightly LoRA-adapted versions of the base model are reasonable "cold starts" for update propagation. We compare seed, checkpoint, and weight-noise calibration in Figure 11.

One specific extension is to learn a linear regression on sets of diverging models and their resulting output variation. This can improve update triage over time and reduce the cost of the τ criterion by shrinking the probe size needed for a robust signal. As we observe new upstream updates and their downstream consequences, we can refine the mapping from representational shift features to consequential output variation. We show early results from this in Figure 1 C. We provide additional details regarding τ's selection in Appendix D.

4.5 Computational Cost

Updates currently trigger retraining and re-evaluation, which incur costs. Compared with retraining, our diagnostic avoids gradient updates entirely: it requires only a single forward pass on a probe dataset—2 and 3 passes for 2- and 3-way comparisons respectively—followed by post-hoc comparisons. Compared with many generative evaluation pipelines, it also avoids autoregressive decoding. Once extracted, geometric comparisons scale linearly $\mathcal{O}(N)$; CKA requires computing and aligning Gram matrices $\mathcal{O}(N^2 \cdot d)$ but is computed on CPU over fixed hidden states. The information sufficiency approach addresses distributional questions about information preservation and is more expensive: PCA reduction ($\mathcal{O}(N \cdot d^2)$), GMM fitting ($\mathcal{O}(N \cdot K \cdot d')$), and sliced Wasserstein ($\mathcal{O}(N \cdot S)$) are all post-GPU. In both cases, the probe set size is the primary lever controlling cost. The advantage is largest when downstream evaluation is long-form, model- or human-graded, or repeated across many candidate updates; for short discriminative evaluations, the diagnostic is best understood as a pre-adoption signal rather than a universal replacement for evaluation. Empirical compute estimates for our experiments are reported in Appendix C; pairwise CKA/JS comparisons took 3–5 minutes per pair, while the GMM sweep took 17–22 minutes per fit.

5 Evaluations

As discussed in Section 1, there are several reasons for model updates. We consider three scenarios: upstream training data interventions, upstream quantization, and safety interventions. In the main body of the paper, we primarily focus on data interventions, i.e., cases where a data deletion or addition has occurred. This provides a constrained testbed for interventions with established consequences on models' learned representations [O'Brien et al., 2025, Deeb and Roger, 2025] that also aligns with real world AI coordination challenges.

Specifically, our interest lies in understanding how downstream applications are impacted by adopting updates. We evaluate **(1)** whether downstream settings are uniformly impacted by these modifications; **(2)** whether representational changes induced by the deletion correlate with performance changes and propagate to downstream models; and **(3)** whether our framework can flag perturbations in downstream M_1' before it is finetuned.

5.1 Experiments

Recall S is our pretraining dataset. Suppose that a subset T induces an upstream intervention, either by removal ($S' = S \setminus T$) or addition ($S' = S \cup T$), producing an updated model $M_1 = f(S')$. Note that while M_1 and M share a common ancestor $a(M)$, they may lack a direct derivation edge in $\mathcal{G}$ (i.e., $(M, M_1) \notin E$), though M_1 still represents a rebased update in the release sequence.

We evaluate the downstream impact of such updates in two settings: (i) *vision* with ResNet models, and (ii) *language*, using LLMs ranging from $\sim 2.5 - 3$B to ~ 7.5B parameters. In each setting, we adapt pretrained models on two distinct tasks. In language, the tasks entail corpora of case law and biology text. In images, we use plane and dog breed image datasets. For each domain, we consider both the reference base model M and an updated model M_1, along with task-adapted variants M' and M_1' via fine-tuning or lightweight adapters. We construct ordered families of upstream models by varying the amount of domain-specific data available during training: moving in one direction corresponds to removing more of that data, while moving in the opposite direction corresponds to adding it back. Practically, the sweep can be treated as sequential updates producing $M_{i=0\ldots n}$, as shown in Figure 1B; for any comparison, we choose a reference point in this sequence as M and compare it to a neighboring or more distant variant M_1. Additional details are included in Appendix B. While data removal and addition are functionally equivalent under this ordering, the reference model for τ changes based on the direction.

Image Experiments We created a classification dataset from two fine-grained vision datasets, the Tsinghua Dogs dataset [Zou et al., 2020] and the FGVC Aircraft dataset [Maji et al., 2013]. Data was preprocessed, augmented [Engstrom et al., 2019], then used to pretrain ResNet-18 models. Both domains differ semantically and in label structure, inducing related but distinct downstream data distributions. Downstream models were fine-tuned on heldout classes from each domain such that each task drew from an overlapping visual manifold while relying on different discriminative features. Mixed ablations were used for baseline comparisons. K-fold cross validation, training seeds, and ablation intervention were varied for a total of 1332 converged models.

LLM Experiments We trained Qwen2.5-3B-parameter LLMs on two independent applications, case law [Naudet and Dolan, 2024] and biology [Li et al., 2024]. These data were used to fine-tune a series of models starting from both original and ablated base checkpoints. Ablation intervention level, fine-tuning domain, and test domain were varied across an 8-way factorial design (representing 84 unique cases across the training trajectory). A second set of experiments used ~ 7.5B parameter LLMs from O'Brien et al. [2025], which pre-trained related models with combinations of data filtering and safety interventions targeting biorisk; we treat these as updates to an unfiltered base model and report results in Figure 6. We report additional robustness experiments in Appendix E, including results across Qwen model scales (0.5B–3B; Figure 12), Pythia models (70M–3B) under FP16, INT8, and NF4 quantization (Figure 13), and safety-relevant circuit-breaker interventions (Figure 14).

The end-to-end procedure is as follows. Given M, M_1, M', and a probe set X, we extract layer-wise hidden-state activations at multiple network layers and compute pairwise similarities between models, aggregated across layers (Appendix B.2). $\mathrm{Sim}(\cdot, \cdot)$ is instantiated either as a geometric distance (Section 4.2, CKA unless otherwise noted) or as a distributional divergence (Section 4.3,

sliced Wasserstein over GMM-fitted embeddings). Prior to fitting, activations are mean-centered and Procrustes-aligned; GMM hyperparameters are selected via grid search (Appendix B). The flagging rule from Equation 1 or 2 is then applied, producing a binary signal indicating whether M' warrants retraining or re-evaluation.

6 Results

We measure change in model outputs as a result of updates by computing the Jensen–Shannon divergence between softmax/posterior distributions produced by model pairs on a fixed probe set, alongside task performance metrics (F1, accuracy, and loss). We use JS-divergence and performance on M_1' as baselines: our methods should flag updates that introduce substantive differences in their outputs, but do so without knowledge of M_1'. In all domains, model updates under targeted deletion exhibit localized rather than global effects, with the loss delta—quantified as the Jensen–Shannon divergence $\Delta\mathcal{L}_{\text{JS}}$ between M' and M_1' outputs—growing when the ablation aligns with the downstream task domain (Figure 4). This confirms that some cases of model drift are not the result of global collapse but rather a localized semantic disruption within the representation manifold, and supports our hypothesis that update adoption may be systematically targeted.

We find that CKA and GMM-based distances effectively track these behavioral shifts, with high correlation between $\text{Sim}(\cdot)$ and output divergence (Appendix Table 3). Direct comparisons between the base model M and its ablated version M_1 often underreport functional impact, remaining within a conservative "no-retrain zone" τ (Figure 8). However, by triangulating these models with a fine-tuned reference M', we can map abstract shifts in representation space to concrete performance degradation (Figure 7). The strong correlation ($r = 0.93$ and $r = 0.88$ for vision and language respectively) between base-level manifold distortion and downstream output divergence shows that a linear mapping can accurately predict functional loss without requiring explicit access to the updated downstream model. Using only M, M_1, M', we can recover a high percentage of the variance in downstream model divergence without ever having to train M_1'.

7 Discussion

Deciding when to adopt upstream updates or how to reconcile their supply-chain-wide impact remains a nascent challenge for the AI ecosystem. This work is intended to formulate model updates as a new problem and explore a possible response to their management and coordination. Our empirical results indicate that representational signals can support principled update triage, allowing sensitivity to vary according to task relevance and organizational risk tolerance, and in some cases more cheaply than existing standards (see 4.5 for details on cost complexity).

Our interests lie in supporting both open and closed model developers. Appendix Table 2 summarizes how our approaches map to deployment settings and actors. For open settings, we introduce a new method for comparing updates to minimize costs in larger eval or retraining paradigms. For closed models, our approach supports future context-aware reporting. By characterizing the relevance of an update to specific downstream representations, providers can communicate the information necessary for responsible decision-making without resorting to uniform, global disclosures. Such targeted signaling may support future auditability and data-deletion obligations while minimizing unnecessary exposure of proprietary details.

In open models, decision-making falls on both upstream model providers *and* downstream developers. Model providers choose to release new models, while developers choose if and when to adopt them. Organizations that are in highly regulated domains, that are less-resourced, or that seek to minimize product instability will adopt less frequently. In domains like medicine, where the expense of additional evaluation can far outstrip the costs of retraining, these unknowns can make adoption unattractive [U.S. Food and Drug Administration, 2024]. Organizations focused on improvements and growth will update regularly to maintain a competitive edge.[5]

[5]Adding to this complexity, some open model licensing place requirements on users to encourage them to adopt upstream modifications. For example, the BigScience OpenRAIL-M License states: *You shall undertake reasonable efforts to use the latest version of the Model* (Section IV.7). The intention with this language is to mitigate risks when upstream modifications such as data deletions or safety interventions are required, though tooling or enforcement remains largely unrealized.

Developers reliant on closed models have limited optionality; the onus of updating falls largely on model providers, who mechanically govern update propagation in API environments. In some cases, timestamped releases and retirement dates are attached to a newly released model (e.g., Google, Anthropic) indicating when downstream users will no longer be able to access it. These dates do not necessarily guarantee stability however—invisible, undisclosed modifications may still impact model performance [Cen et al., 2025]—but they do give developers the ability to choose when to switch models. Here, we believe there's more that should be done to balance the quality of information shared with downstream developers with the other constraints faced by model developers (like IP).

Closed model providers face the same challenges of update management internally. Even within large companies that maintain multiple internal models, deciding when and how to propagate updates across interconnected applications requires repeated cycles of training, evaluation, and deployment. These providers also fine-tune models on contractual basis. Deciding how to propagate model updates internally or inform the orgs using them are relatively open questions.

As AI deployment has matured, the median downstream developer has moved further away from their model's internals. Proliferating fine-tuning and model-as-a-service platforms escalate the complexity of update management, yet abstract away many of the underlying mechanics that would otherwise inform downstream risk and adoption decisions. In these settings, tools to determine when an update is necessary—along with relevant disclosures for users—are needed.

Closed model providers possessing access to internal activations can compute such diagnostics on behalf of their users, giving them a mechanism to issue algorithmic update flags while reducing proprietary information disclosure. Downstream developers navigate upstream changes with little guidance, often under partial or asymmetrical information. In effect, updates remain 'black-box' transitions where the impacts on downstream dependencies remain largely opaque until after adoption. Finally, JS-divergence on output logprobs (which many providers include with API access) is a reasonable, if expensive, secondary flag (requiring full autoregressive generation across test cases).

Calibrating τ to the Adopter In our framework, τ is not a single, fixed quantity. Rather, it is a configurable surface that adopters can tune to their own constraints. Several parameters allow τ to be tailored to the needs of an adopter. Which similarity metric to use is one such parameter: geometric comparisons (e.g., CKA) capture structural alignment, while information-theoretic measures (e.g., JS, Wasserstein) capture distributional shift. The probe used to produce τ is another. Smaller probes are cheaper but noisier, while larger sets stabilize the bound at additional cost (see Figure 11). The comparison type (2-way versus 3-way) reflects whether the adopter is using the model directly or fine-tuning on top of it, with the 3-way variant additionally accounting for downstream task adaptation. The *method of replica production* shapes what variation τ treats as noise: replicas drawn from different random seeds, intermediate checkpoints, LoRA adaptations, or small weight perturbations each define a different acceptable thresholds, and therefore a different intervention criterion. These "knobs" allow adopters to trade sensitivity for cost and tailor τ to organizational risk tolerance rather than committing to a one-size-fits-all threshold.

7.1 Limitations

While we characterize uncertainty across different selections, results are still sensitive to what data is used to probe the model. Threshold calibration (τ) is empirical, and the uncertainty band may not transfer cleanly across model families or deployment settings without recalibration. GMMs also assume underlying data is Gaussian–which may not hold in LLMs. These methods must be evaluated at scale under more complex interventions. Our evaluations also focus on discrete update pairs; how well these methods scale over time—across longer sequences of updates—is an open question.

References

J. Ashkenas. Why semantic versioning isn't. GitHub Gist, Aug. 2014. URL `https://gist.github.com/jashkenas/cbd2b088e20279ae2c8e`. Created August 29, 2014; accessed May 26, 2026.

G. Bansal, B. Nushi, E. Kamar, D. S. Weld, W. S. Lasecki, and E. Horvitz. Updates in human-AI teams: Understanding and addressing the performance/compatibility tradeoff. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 2429–2437, 2019.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

L. Bereska and E. Gavves. Mechanistic interpretability for AI safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

R. Bommasani, D. Soylu, T. I. Liao, K. A. Creel, and P. Liang. Ecosystem graphs: The social footprint of foundation models, 2023. URL https://arxiv.org/abs/2303.15772.

S. H. Cen, A. Ilyas, H. Driss, C. Park, A. Hopkins, C. Podimata, et al. Large-scale, longitudinal study of large language models during the 2024 US election season. *arXiv preprint arXiv:2509.18446*, 2025.

M. Darrin, P. Formont, I. Ayed, J. C. Cheung, and P. Piantanida. When is an embedding model more promising than another? *Advances in Neural Information Processing Systems*, 37:68330–68379, 2024.

A. Decan and T. Mens. What do package dependencies tell us about semantic versioning? *IEEE Transactions on Software Engineering*, 47(6):1226–1240, 2019.

A. Deeb and F. Roger. Do unlearning methods remove information from language model weights?, 2025. URL https://arxiv.org/abs/2410.08827.

L. Engstrom, A. Ilyas, H. Salman, S. Santurkar, and D. Tsipras. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

X. Han, Z. Wang, B. Zhao, B. Zhang, J. Li, D. Borth, R. Yu, H. Maron, Y. Ye, L. Yin, et al. A survey of weight space learning: Understanding, representation, and generation. *arXiv preprint arXiv:2603.10090*, 2026.

A. Hopkins, S. H. Cen, A. Ilyas, I. Struckman, L. Videgaray, and A. Madry. AI supply chains: An emerging ecosystem of AI actors, products, and services. *arXiv preprint arXiv:2504.20185*, 2025a.

A. Hopkins, I. Struckman, K. Klyman, and S. S. Silbey. Recourse, repair, reparation, & prevention: A stakeholder analysis of AI supply chains. In *Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency*, pages 209–227, 2025b.

E. Horwitz, B. Cavia, J. Kahana, and Y. Hoshen. Learning on model weights using tree experts, 2025a. URL https://arxiv.org/abs/2410.13569.

E. Horwitz, N. Kurer, J. Kahana, L. Amar, and Y. Hoshen. We should chart an atlas of all the world's models, 2025b. URL https://arxiv.org/abs/2503.10633.

M. Huh, B. Cheung, T. Wang, and P. Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024.

J. Kahana, E. Horwitz, I. Shuval, and Y. Hoshen. Deep linear probe generators for weight space learning. *arXiv preprint arXiv:2410.10811*, 2024.

K. Lee, A. F. Cooper, and J. Grimmelmann. Talkin' 'bout AI generation: Copyright and the generative-AI supply chain. *arXiv preprint arXiv:2309.08133*, 2023.

N. Li, A. Pan, A. Gopal, S. Yue, D. Berrios, A. Gatti, J. D. Li, A.-K. Dombrowski, S. Goel, L. Phan, G. Mukobi, N. Helm-Burger, R. Lababidi, L. Justen, A. B. Liu, M. Chen, I. Barrass, O. Zhang, X. Zhu, R. Tamirisa, B. Bharathi, A. Khoja, Z. Zhao, A. Herbert-Voss, C. B. Breuer, S. Marks, O. Patel, A. Zou, M. Mazeika, Z. Wang, P. Oswal, W. Liu, A. A. Hunt, J. Tienken-Harder, K. Y. Shih, K. Talley, J. Guan, R. Kaplan, I. Steneker, D. Campbell, B. Jokubaitis, A. Levinson, J. Wang, W. Qian, K. K. Karmakar, S. Basart, S. Fitz, M. Levine, P. Kumaraguru, U. Tupakula, V. Varadharajan, Y. Shoshitaishvili, J. Ba, K. M. Esvelt, A. Wang, and D. Hendrycks. The wmdp benchmark: Measuring and reducing malicious use with unlearning, 2024.

S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

L. B. Naudet and T. Dolan. The case-law, centralizing legal decisions for better use. `https://huggingface.co/datasets/HFforLegal/case-law`, 2024.

K. O'Brien, S. Casper, Q. Anthony, T. Korbak, R. Kirk, X. Davies, I. Mishra, G. Irving, Y. Gal, and S. Biderman. Deep ignorance: Filtering pretraining data builds tamper-resistant safeguards into open-weight llms. *arXiv preprint arXiv:2508.06601*, 2025.

L. Ochoa, T. Degueule, J.-R. Falleri, and J. Vinju. Breaking bad? semantic versioning and impact of breaking changes in maven central: An external and differentiated replication study. *Empirical Software Engineering*, 27(3):61, 2022.

C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.

D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.

T. Preston-Werner. Semantic versioning 1.0.0. Semantic Versioning, 2011. URL `https://semver.org/spec/v1.0.0.html`. Accessed May 26, 2026.

J. Rushby. *Formal methods and the certification of critical systems*, volume 37. SRI International, Computer Science Laboratory, 1993.

U.S. Food and Drug Administration. Marketing submission recommendations for a predetermined change control plan for artificial intelligence-enabled device software functions. Technical report, U.S. Food and Drug Administration, December 2024. URL `https://www.fda.gov/regulatory-information/search-fda-guidance-documents/marketing-submission-recommendations-predetermined-change-control-plan-artificial-intelligence` Final Guidance. Accessed September 21, 2025.

D.-N. Zou, S.-H. Zhang, T.-J. Mu, and M. Zhang. A new dataset of dog breed images and a benchmark for finegrained classification. *Computational Visual Media*, 6(4):477–487, 2020.

A Information Sufficiency

Sufficiency formalizes when a model A retains all necessary information from another model B. We redefine sufficiency in the context of model updates, stating that M_1 is sufficient for M, denoted as $M_1 \succeq_S M$, if and only if there exists a stochastic transformation $\mathcal{M}$ (a Markov kernel) such that:

$$E \left\| \mathcal{M} \circ P_{M_1|X} - P_{M|X} \right\|_{\text{TV}} = 0. \tag{7}$$

Here, $P_{M|X}$ and $P_{M_1|X}$ denote the distributions of embeddings produced by models M and M_1, respectively, when evaluated on the input distribution X. That is, $P_{M|X} = \left\{ P(H_M^0(X)), P(H_M^1(X)), \ldots, P(H_M^k(X)) \right\}$, with the same notation applying to $P_{M_1|X}$. These are the distributions over representations at each relevant layer.

The transformation $\mathcal{M}$ maps representations from M_1 back to M, ensuring that no information is lost. A divergence measure, like total variation (TV) $\| \cdot \|_{\text{TV}}$ or Wasserstein distance $\| \cdot \|_{\text{EMD}}$ measures the discrepancy between two distributions. Note that while these measures are somewhat interchangeable, Total Variation (TV) provides an upper bound on probability mass shifts, Wasserstein distance (EMD) captures shifts in embedding structure beyond first-order statistics, while Jensen-Shannon (JS) is symmetric. We empirically test each of these in our ResNet experiments.

If $M' \succeq_S M$, then any model relying on facts learned from M should function equivalently when using representations from M', thereby eliminating the need for retraining.[6]

[6]When classification labels Y are available, sufficiency implies preservation of Bayes risk, as shown in [Darrin et al., 2024].

Under this definition of sufficiency, equivalence implies that if M_1 is sufficient for M, then no task-specific retraining is needed—the downstream behavior is preserved. Conversely, if sufficiency fails, it indicates that representational changes may impact downstream compatibility, necessitating adaptation.

While the notion of *sufficiency* requires that the updated model's distribution $P_{M_1|X}$ can be transformed exactly into the original model's distribution $P_{M|X}$ via a Markov kernel $\mathcal{M}$, checking this condition is intractable in practice. To overcome this, Darrin et al. [2024] introduce a relaxation termed information sufficiency. Intuitively, information sufficiency measures how much the uncertainty about the embeddings produced by M_1 is reduced by knowing the embeddings from M.

If M_1 captures the downstream data X in a manner similar to M, then knowing $M(X)$ should allow us to predict $M_1(X)$. Then, information sufficiency from M to M_1 is:

$$\text{IS}(M \to M_1) \triangleq \mathcal{H}(M_1(X)) - \mathcal{H}(M_1(X) \mid M(X)),$$

where $\mathcal{H}(M_1(X))$ is the entropy (or average uncertainty) of the embeddings from M_1 on inputs X, and $\mathcal{H}(M_1(X) \mid M(X))$ is the conditional entropy of those embeddings given the embeddings from M. This is a standard decomposition of mutual information $I(M(X); M_1(X))$. In the three-way setting, the conditioning term uses M' in place of M_1, assessing whether M_1 preserves the information that the downstream model M' relies on from M. Given two models, e.g., pick $A, B \in \{M, M', M_1\}$:

$$\text{Sim}(A, B) = D_{\text{div}}\left(P(H_A^k \mid X) \,\|\, P(H_B^k \mid X)\right), \tag{8}$$

where D_{div} is our selected divergence measure. Then, we evaluate the alignment between M, M', and M_1 with one of our criteria from 1 and 2, conditioned on a shared input X using our original criteria. This formulation extends information sufficiency to incorporate a second behavioral proxy for downstream propagation, allowing us to detect when representational changes in the upstream model might misalign with the downstream model's learned behavior. If information sufficiency fails, it implies information has been lost or altered for X in the update step, suggesting retraining will lead to a distribution shift.

B Experiments

We estimate representation shift using the *same* fixed benchmark set $\mathcal{X}$ for *every* model and comparison. All models run with identical preprocessing and batching; seeds are fixed across Python/NumPy/torch and deterministic backends are enabled.

B.1 LLM Experiments

We construct model variants by adding or removing domain-specific data from pretraining at varying levels; these are structurally equivalent operations—adding domain data to a base model trained without it induces the same representational shift as removing it from one trained with it. Let $\mathcal{D}_{\text{base}}$ denote the original pretraining data, partitioned into domain-specific subsets $\mathcal{D}_{\text{bio}}$ (biology/medical) and $\mathcal{D}_{\text{law}}$ (legal). We create ablated variants:

$$\mathcal{D}_{\text{ablated}}^{(p)} = \mathcal{D}_{\text{base}} \setminus (p \cdot \mathcal{D}_{\text{domain}}) \tag{9}$$

where $p \in \{5\%, 10\%, 25\%, 50\%, 100\%\}$ and domain $\in \{\text{bio}, \text{law}\}$, yielding two families: one retaining only LAW data and one retaining only BIO data. We additionally evaluate robustness across Qwen model scales (0.5B–3B) and quantization levels (FP16, INT8, NF4), and test on safety-relevant updates via circuit-breaker interventions; results are reported in Appendix E.

B.2 Evaluation

For each fine-tuned LLM, we compute perplexity (via cross-entropy loss) on held-out test sets from both domains. The key metric is the **loss delta**:

$$\Delta L = L_{\text{FT}(M_{\text{abl}})} - L_{\text{FT}(M_{\text{orig}})} \tag{10}$$

where $L_{\mathrm{FT}(M)}$ denotes the loss of the model fine-tuned from base M. Positive ΔL indicates degradation relative to the original.

Aggregation Strategies Aggregations for image models included the following:

$$\mathrm{Agg}(k) = \begin{cases} \max_k, & \text{captures worst-layer discrepancy} \\ \min_k, & \text{best-layer match} \\ \frac{1}{K}\sum_k, & \text{average over all layers} \\ \frac{1}{3}\sum_{k=4}^{6}, & \text{tail3 (last three layers)} \end{cases}$$

Similar strategy, though not in number of layers, was taken for the LLM experiments.

B.3 Image Experiments

We created a classification dataset S (recall base model $M = f(S)$) from balanced subsets of two fine-grained visual classification datasets, the Tsinghua Dogs dataset [Zou et al., 2020] and the FGVC Aircraft dataset [Maji et al., 2013]. S contains 3000 total images with labels evenly distributed between 30 classes (15 from each dataset). After pre-processing and transformations, S contains 24000 total images (8 versions of each of the originals).

We use the 3,000-image validation sets V_1 or V_2. For every model we extract activations $\{h_i^k\}_{i,k}$ at the output of the convolutional layer and each residual block ($k = 1, \ldots, 5$) and the penultimate global-average-pooled layer ($k = 6$).

The downstream task of M' is distinguishing between different dog breeds, while the downstream task of M'' involves differentiating plane manufacturers. These downstream tasks are intentionally designed to be fine-grained and visually similar within their respective domains, promoting clear yet subtle distinctions in learned features. The upstream task, by contrast, is more general, simultaneously covering both dog breeds and plane manufacturers. This structure offers several advantages: it explicitly isolates the influence of specific subsets of data deletions, making causal relationships between upstream modifications and downstream performance more evident. Moreover, by carefully controlling the overlap and differences in data subsets, the dataset design enables precise evaluation of whether certain upstream deletions impact only relevant downstream tasks or inadvertently affect unrelated ones.

To finetune, we used two other subsets from those datasets to define two distinct downstream train datasets, R_1 and R_2 (where $M' = g(M, R_1)$ and $M'' = g(M, R_2)$). Each R_x contains 750 images with labels evenly distributed between 15 classes (either dog breeds or plane manufacturers). After pre-processing and transformations (see Appendix B), each R_x contains 6000 total images (8 versions of each of the originals).

This makes it much more straightforward to isolate deletion impacts than in practical scenarios, where upstream datasets often have subtler and less explicitly segmented categories. As a result, we can more easily evaluate our retraining thresholds. It thus provides a strong foundational model to rigorously explore and clearly demonstrate the relationships between data deletions upstream and their downstream implications.

We adopt two data deletion strategies to produce S' (where $S' = S \setminus T$). The first method of data deletion was by class, where all images from a given class were removed from S. The second method of data deletion was by percentage, where a percentage of images from all relevant classes were removed from S. When randomly selecting data to delete (T), we selected in three subsets: only from classes relevant to the dog breed classification task, only from classes relevant to the plane manufacturer classification task, and from all classes. Although this vision-only, small-scale setting is simplified relative to real supply chains, it offers a tractable sandbox in which to develop and validate principled criteria for update propagation.

Finally, for evaluation, we produce two 3,000-image test sets E_1, E_2, and V_1,V_2 for each downstream task. E_x is used only to evaluate downstream task performance via accuracy and F1. V_x is used for every similarity metric we consider and covers the downstream task distribution. It's critical that downstream developers have a benchmark set that is an approximate and reasonable coverage of their downstream application.

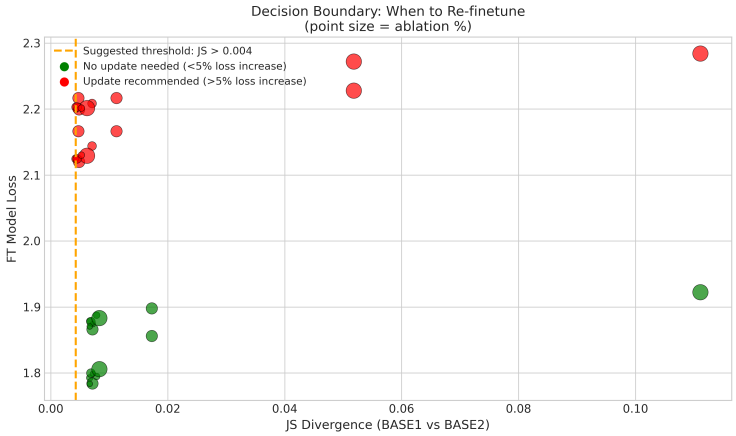

Figure 5: Decision boundary for triggering downstream re-finetuning based on upstream divergence. Each point corresponds to a model update, with the x-axis showing the Jensen–Shannon divergence between the base model and its ablated version, and the y-axis showing the validation loss of the fine-tuned model. Point size indicates the fraction of upstream data removed. Green points denote updates that incur less than a 5% increase in downstream loss (no re-finetuning required), while red points denote updates exceeding this threshold (re-finetuning recommended). The dashed vertical line indicates the learned decision threshold on JS divergence, illustrating that upstream representational shift provides a reliable, task-conditioned signal for predicting when downstream performance degradation becomes operationally significant.

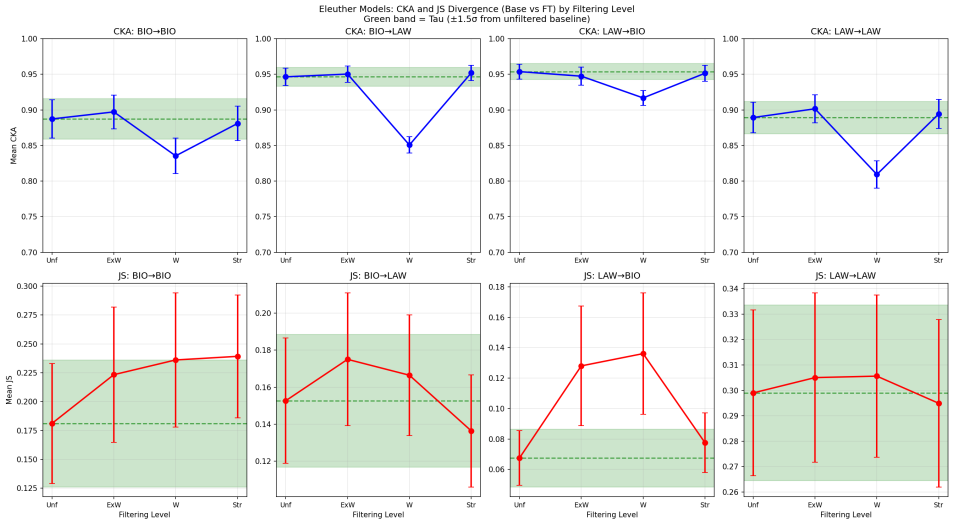

Figure 6: Effect of upstream filtering strength on representational similarity and output divergence across domains from O'Brien et al. [2025]. Top row: Mean CKA similarity between base and fine-tuned models under increasing filtering levels (Unfiltered, Extra-Weak, Weak, Strong) for all combinations of pretraining domain (BIO, LAW) and fine-tuning domain. Bottom row: Corresponding Jensen–Shannon (JS) divergences between model outputs. Green bands indicate the baseline tolerance region τ ($\pm 1.5\sigma$ from the unfiltered condition), and dashed lines mark the baseline mean.

C Compute Estimate

We report estimated GPU-hours for all experiments contributing to the findings reported in this paper, including model training, analysis pipelines, and benchmark evaluations. Estimates are based on observed per-fit timings where available (the GMM bio sweep) and on script-derived hyperparameter counts elsewhere. We use NVIDIA A100 as the baseline; for other hardware, H100 reduces wall-clock by roughly $2\times$ and V100 increases it by roughly $2\times$ for the model sizes considered.

Category	Units	Per-unit (A100)	A100-hr
LLM training			
Qwen continued-pretrain (canonical M + 10 ablations)	11	25–40 min	5–7
Qwen seed retrains for τ (15 seeds, 7,500 steps each) †	15	$\sim$3 hr	$\sim$45
Qwen fine-tune layer (bio_ft, law_ft on canonical M_1's)	$\sim$66	5–10 min	6–11
Pythia fine-tune (deep-ignorance variants $\times$ FT)	32	$\sim$30 min	$\sim$16
Analysis: GMM-based information sufficiency			
GMM bio sweep, canonical fits + diagnostic tests 1–3	28 fits	$\sim$17 min	$\sim$10
WMDP-bio MCQ benchmark (1,273 questions $\times$ checkpoint)	163	$\sim$1 min	$\sim$3
Analysis: distance-based (CKA, JS-on-logprobs)			
Pairwise comparisons (mega suite + interventions + ablations)	$\sim$300 pairs	3–5 min	20–25
τ pipeline / probe-resampling for CKA	50–80 jobs	$\sim$10 min	10–15
Vision precedent (information sufficiency, dog/plane classification)			
Vision model training + GMM fitting + comparisons	50–80 jobs	$\sim$30 min	30–40
Total (range)			**145–185**

Table 1: Estimated GPU-hours per experimental category. Unit counts come from manifest files, job-list scripts, and result-directory inventories. Per-unit times for the GMM bio sweep are observed (mean of 12 timings: 17–22 min, range $\pm$15%); other per-unit times are extrapolated from the same hyperparameters.

C.1 Total compute and hardware ratios

The point estimate is approximately **150 A100 GPU-hours**.

Qwen seed retrains ($\sim$45 A100-hr). Fifteen replicas of Qwen2.5-1.5B continued-pretrained for 7,500 steps each. Per-step time on a single A100 at batch size 128 with 2,048-token sequences and FlashAttention is approximately 1.5 seconds; this gives $\sim$3 hours per seed and 45 hours total. If the actual training was distributed across 4–8 GPUs with DDP, the total GPU-hours could rise to 60–80 due to communication overhead, although wall-clock drops correspondingly.

Vision pipeline (30–40 A100-hr). Per-training times for the small vision models (e.g., ResNet-50, ConvNeXt) are short individually but accumulate; we estimate $\sim$30 minutes per training including pretrain and 1–2 fine-tunes.

Analysis estimates. GMM bio-sweep timings are directly observed ($\pm$15% across 12 fits). WMDP-bio evaluation cost was estimated by counting model size, prompt length, and forward-pass cost; CKA/JS pair counts were extracted from the per-pair result TSVs and job-list manifests in the analysis tree.

C.2 What is not counted

- **Original Qwen2.5-1.5B pretraining**, performed by the Qwen team and likely on the order of 10^6 GPU-hours. Our work continues pretraining from this checkpoint and consumes only the additional compute reported above.
- **EleutherAI Pythia base models and their deep-ignorance variants**, which we use directly without retraining.
- **Failed runs, debugging iterations, and hyperparameter search**. The reported numbers reflect the final, successful run-set.

- **Local data preprocessing and analysis** (CPU-bound; negligible on the GPU-hour scale).

C.3 Reproducibility

The smallest end-to-end reproduction of the GMM-based information-sufficiency analysis (the central method contribution of §4.3) requires:

- Approximately **2 A100-hours** of training (canonical M plus 5 representative ablations, no seed retrains for τ);
- Approximately **2 A100-hours** of analysis (GMM fitting, Procrustes alignment, τ derivation, and benchmark evaluation).

A full reproduction including the noise-floor seed pool and the diagnostic tests (§**??**) requires the full $\sim$150 A100-hours reported in Table 1.

D Picking τ

A τ is reasonable if (1) it is built from a comparison set that captures unimportant variation (i.e., what you would prefer to ignore), and (2) it is stable such that re-sampling doesn't lead to significant uncertainty or variance in its boundaries.

To this end, replicas were obtained either by training from scratch with different random seeds and data slices, via controlled perturbations to the model such as LoRA adaptations or intermediate checkpoints, or from different training steps.

We also learn a linear regression model as follows.

Let the target of our regression be the downstream divergence:

$$\Delta\mathcal{L}_{\text{JS}} = JS(M_1, M_1')$$

We predict this using a linear combination of upstream metrics:

$$\Delta\mathcal{L}_{\text{JS}} = \alpha \cdot Sim(M, M') + \beta \cdot Sim(M, M_1) + \gamma.$$

This can be expanded to account for additional metrics, which provide more signal. For example, the actual JS-divergence between upstream models and/or the original fine-tuned variant offers additional features that improve regression.

E Additional Results

Access	Deployment	Method	Actor
System Improvement (2 & 3-way geometric comparison)			
Open	Direct	Geometric	Developer
Open	Fine-tuning	Geometric	Developer
Closed	Both	Geometric	Provider / 3rd party
Product Reliability (2 & 3-way geometric comparison)			
Open	Direct	Geometric	Developer
Open	Fine-tuning	Geometric	Developer
Closed	API	Output JS div.	Developer
Closed	Both	Geometric	Provider / 3rd party
Information Preservation (2 & 3-way Information Sufficiency)			
Open	Direct	IS (GMM)	Developer
Open	Fine-tuning	IS (GMM)	Developer
Closed	Both	IS (GMM)	Provider / 3rd party

Table 2: Framework methods by use case and deployment setting. All comparisons use calibrated τ. Information Preservation covers settings where the nature of the change matters—such as data deletion or compliance—rather than its magnitude alone. In closed settings, comparisons can be run by the provider or a trusted third party for internal management or external targeted disclosures.

Table 3: Correlation Analysis: Can Base Model Metrics Predict Re-finetuning Need?

Comparison	r	p-value	n	Sig.
JS(BASE_1, BASE_2) vs JS(FT_1, FT_2)	0.880	<0.0001	33	✓
JS(BASE_1, BASE_2) vs Loss(FT_2) [all]	0.168	0.3141	38	
JS(BASE_1, BASE_2) vs Loss(FT_2) [bio]	0.572	0.0084	20	✓
JS(BASE_1, BASE_2) vs Loss(FT_2) [law]	0.761	0.0002	18	✓
CKA(BASE_1, BASE_2) vs Loss(FT_2)	0.952	0.0000	10	✓
JS(FT_1, FT_2) vs Loss(FT_2)	-0.041	0.7357	70	

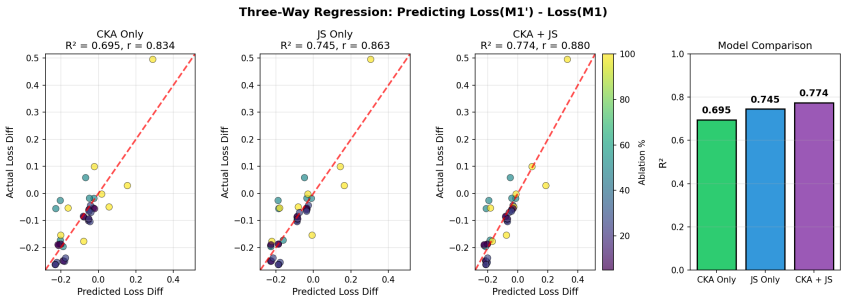

Figure 7: Three-way regression predicting downstream loss change. Scatter plots show predicted versus actual loss difference $\Delta L = L(M_1') - L(M_1)$ using (left) CKA features only, (middle) JS divergence features only, and (right) their combination. Point color and size indicate the fraction of upstream data ablated. The dashed line denotes the identity line. Combining geometric (CKA) and information-theoretic (JS) signals yields the highest explanatory power ($R^2 = 0.774$, $r = 0.88$), outperforming either metric alone. Right: comparison of R^2 across feature sets, demonstrating that triangulating the base model, its ablated variant, and the fine-tuned model provides the most accurate prediction of downstream performance degradation.

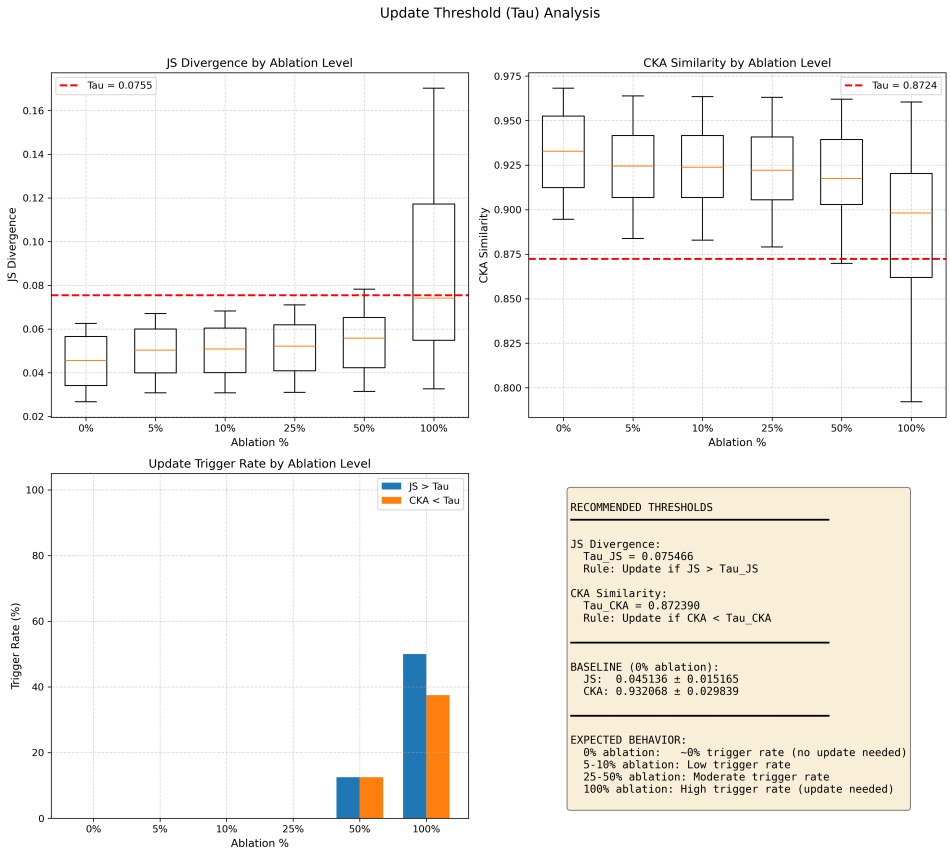

Figure 8: Update Threshold (τ) Analysis for JS Divergence and CKA Similarity. Top: Distribution of Jensen–Shannon (JS) divergence and CKA similarity between fine-tuned models and ablated bases as a function of the fraction of upstream data removed. Dashed lines indicate the learned update thresholds τ_{JS} and τ_{CKA}, above (JS) or below (CKA) which an update is flagged as potentially consequential. Bottom left: Fraction of model pairs exceeding the respective thresholds (trigger rate) at each ablation level, showing low false-positive rates at small deletions and rapidly increasing sensitivity for large, in-domain removals. Bottom right: Summary of recommended thresholds and baseline statistics. Together, these results illustrate that task-aligned deletions cross representational and output-divergence thresholds in a predictable manner, enabling principled update triage based on geometric and information-theoretic criteria.

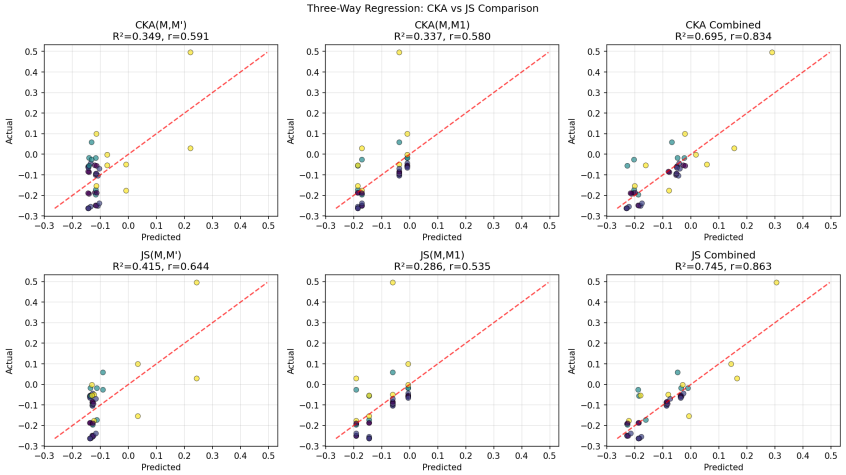

Figure 9: Three-way regression with decomposed geometric and information-theoretic features. Top row: Prediction of downstream loss change using CKA-based distances between the base and ablated models $\mathrm{CKA}(M, M')$, between the base and fine-tuned model $\mathrm{CKA}(M, M_1)$, and their combination. Bottom row: Corresponding regressions using Jensen–Shannon divergences $\mathrm{JS}(M, M')$, $\mathrm{JS}(M, M_1)$, and their combination. Each panel shows predicted versus actual loss difference $\Delta L = L(M_1') - L(M_1)$, with the dashed line indicating the identity. Combining both base-to-base and base-to-fine-tuned comparisons substantially improves predictive power, with the joint JS model achieving the highest fit ($R^2 = 0.745$, $r = 0.863$) and the joint CKA model also showing strong performance ($R^2 = 0.695$, $r = 0.834$). These results confirm that triangulating the base, ablated, and fine-tuned models captures both magnitude and task alignment of representational shift, yielding accurate prediction of downstream performance degradation.

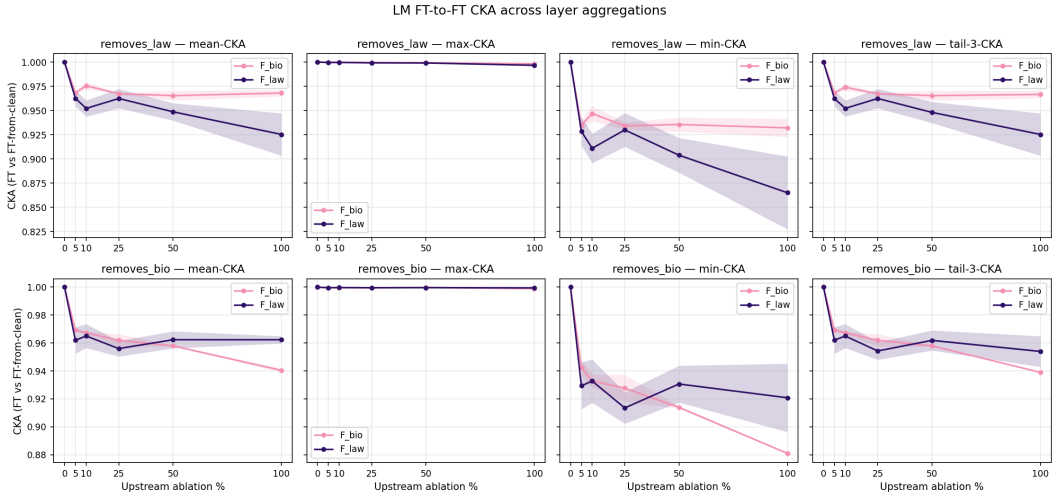

Figure 10: LM fine-tuned-to-fine-tuned CKA across layer aggregation strategies. Each panel shows CKA similarity between fine-tuned models derived from the original versus ablated base checkpoint, as a function of upstream ablation percentage. Top row: models with law data removed from pretraining; bottom row: models with bio data removed. Columns correspond to four aggregation strategies: mean, max, min, and tail-3 (last three layers). Pink lines denote bio fine-tuned models (F_{bio}); blue lines denote law fine-tuned models (F_{law}). In-domain ablations (e.g., removing bio data for bio-fine-tuned models) consistently produce steeper representational divergence than out-of-domain ablations.

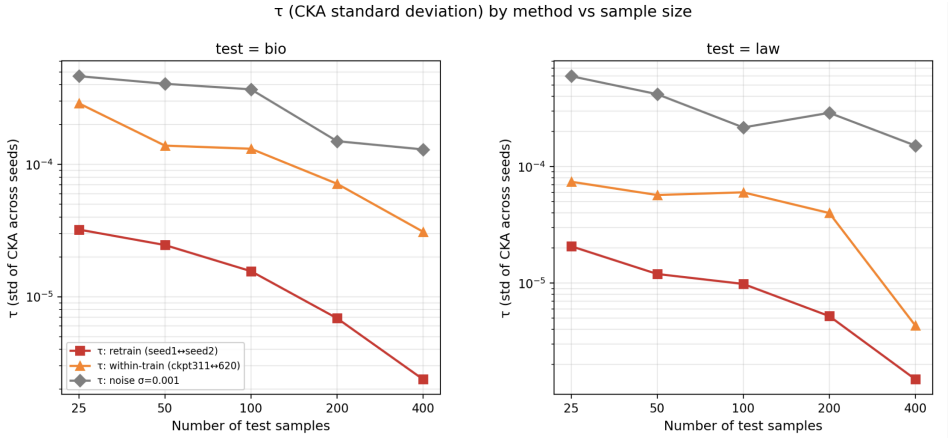

Figure 11: τ calibration stability by method and probe set size. Figures show the standard deviation of CKA across seeds as a function of probe set size (25–400 samples), for bio (left) and law (right) downstream domains, under three calibration strategies: retraining with different random seeds (red), comparing within-training checkpoints (orange), and adding small Gaussian noise perturbations to the weights (grey). All three strategies produce decreasing cross-seed variability as sample size increases. The y-axis is log-scaled and spans 10^{-5} to 10^{-4}; even at the smallest probe sizes, CKA standard deviations remain small in absolute terms. Replica-based calibration (red) is the most stable across both domains, followed by checkpoints.

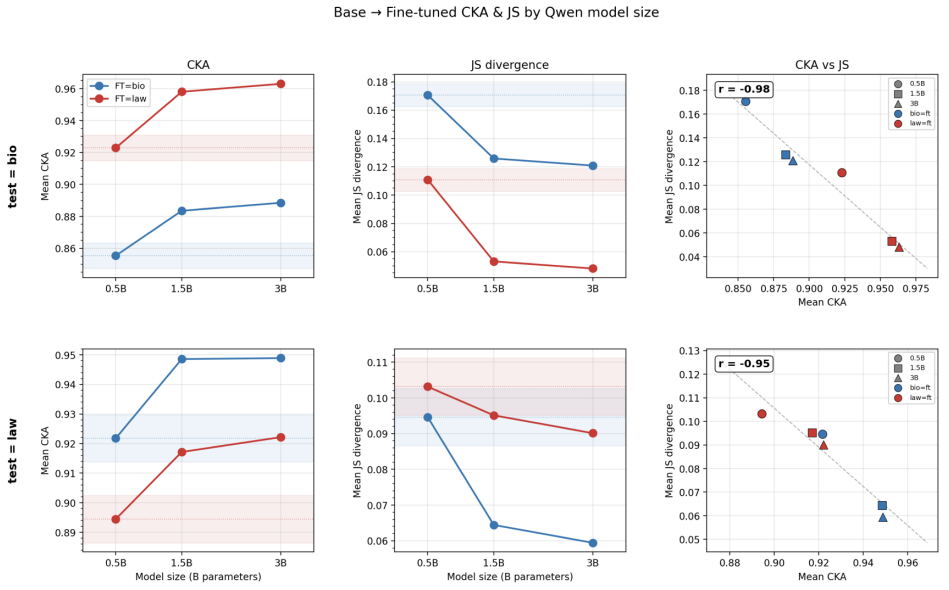

Figure 12: CKA similarity and JS divergence between base and fine-tuned Qwen models across three model sizes (0.5B, 1.5B, 3B), for bio (top) and law (bottom) downstream domains. Across both fine-tuning conditions, larger models show higher CKA and lower JS divergence, indicating that representations in larger models are less perturbed by fine-tuning on a fixed dataset. The framework preserves domain-specific structure across scales, and CKA and JS divergence remain strongly correlated ($r = -0.98$ and $r = -0.95$ respectively).

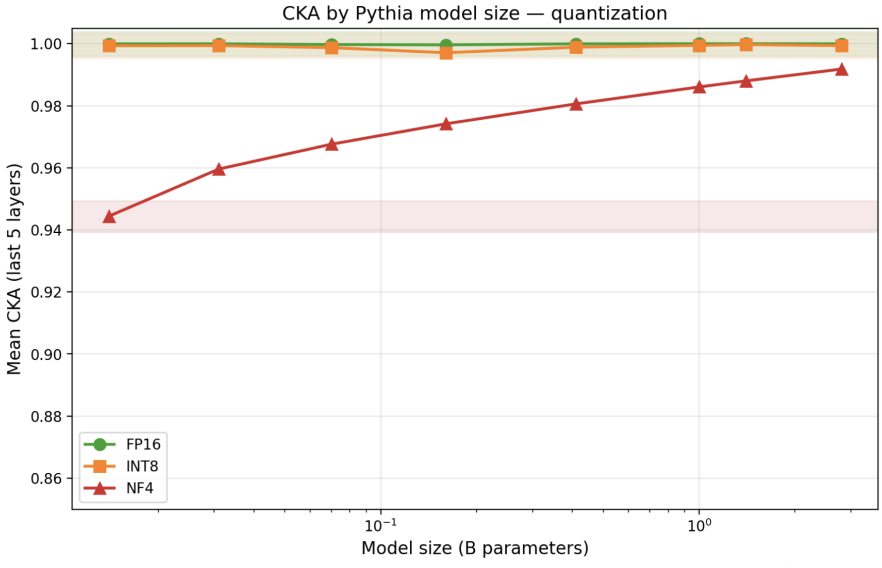

Figure 13: Mean CKA similarity (averaged over the last 5 layers) between base and quantized Pythia models as a function of model size (70M–3B), under three quantization levels: FP16 (green), INT8 (orange), and NF4 (red). FP16 and INT8 remain extremely close to 1.0 across all model sizes, indicating negligible representational change. For NF4, CKA drops for small models but improves monotonically with scale, approaching 1.0 at 3B parameters, suggesting larger models absorb aggressive quantization with less representational disruption.

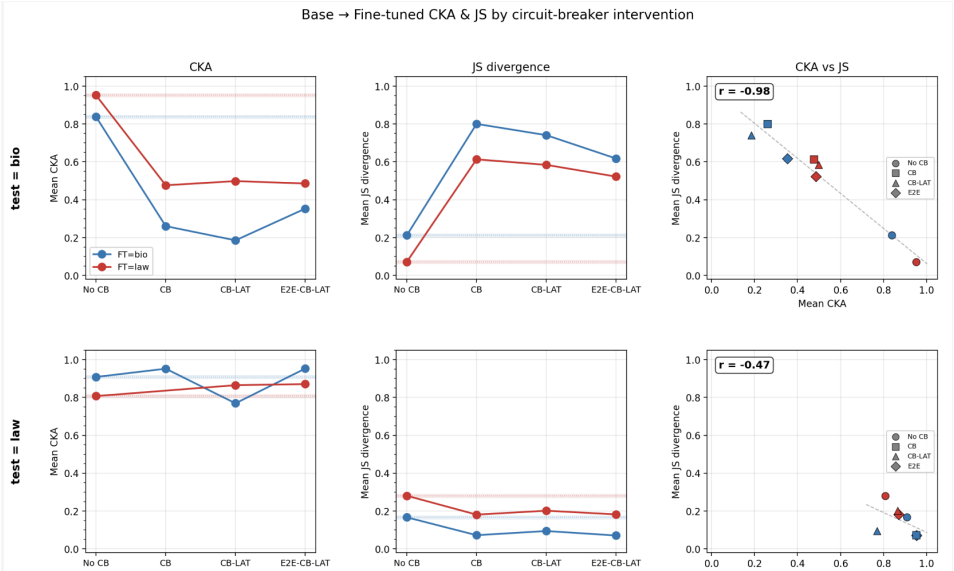

Figure 14: CKA similarity and JS divergence between base and fine-tuned models under progressively intensive circuit-breaker safety interventions (No CB, CB, CB-LAT, E2E-CB-LAT), for bio-risk (top) and law (bottom) fine-tuning domains. Bio-risk fine-tuning representations are significantly perturbed by circuit-breaker interventions, while law fine-tuning remains largely unchanged. CKA and JS divergence are strongly correlated across intervention types ($r = -0.98$ for bio, $r = -0.47$ for law), illustrating that our framework captures the representational disruption introduced by safety interventions in domain-relevant settings.