Training as a Formation

Kenneth Pernyér, Reflective Labs
Stockholm, Sweden
kenneth.pernyer@reflective.se · reflective.se · ORCID 0009-0006-4543-1156

The previous paper in this series argued that closed-form analytics satisfy my substrate's invariants for free, and that fitted models satisfy them only relative to a pinned artifact. This paper takes the other side. It describes Crucible, the extension that owns everything requiring a training loop — adaptive neuro-fuzzy systems fitted by automatic differentiation, random forests and decision trees fitted by classical optimization — and asks what it takes for a fitted model to be a legitimate participant in a governed decision rather than an opaque dependency. My answer has two parts. First, the artifact must be a fact: a trained model enters the context with its dataset, its hyperparameters, its seed and its evaluation attached, so that *which model decided this* has an answer with the same provenance discipline as any other fact. Second, the pipeline that produces it should be a Formation and not a script — the stages of a training run are already Suggestor-shaped, and running them to a fixed point makes retraining a convergent process rather than a scheduled job. I have built the first and designed the second. The pipeline runs from a command line today and lifts into a Formation when a real retrain trigger pulls; I say so plainly rather than describing the design as though it were deployed.

1. Introduction

A fitted model in a governed system is an awkward object. Everything else in the context is a fact with provenance — who proposed it, from what, under which policy — and the model is a binary blob that appeared from a training run somebody did, encoding statistical regularities of a dataset that may no longer exist. When its prediction is used in a decision, the honest answer to *why?* is a shrug and a file path.

That awkwardness isn't an argument against fitting. Some things must be fit: the boundary between a satisfactory and an unsatisfactory loan applicant isn't a rule anyone can write down, and the attempt to write it down is how organizations end up encoding a decade-old sample as policy. It's an argument for treating the artifact with the same seriousness as everything else the system believes.

Crucible is where fitting happens. Its counterpart holds the closed-form packs and never fits (Pernyér, 2025a); Crucible owns training loops, artifacts, a registry, evaluation, monitoring and deployment, and never owns expert rules. The line is drawn at whether parameters arrive with the question or come from a stored artifact, and that's the line the substrate cares about (Pernyér, 2025b).

2. What must be fit

The adaptive neuro-fuzzy case makes the boundary concrete rather than doctrinal. The inference procedure is identical to the one described in the preceding paper (Pernyér, 2025c) — linguistic variables, rules, firing strengths. What changes is where the membership-function parameters come from: an expert's judgement, or a gradient. The first is closed-form and its provenance is a person; the second is fitted and its provenance is a dataset. A system that treats these

Family	How it's fitted
Adaptive neuro-fuzzy	Membership-function parameters by automatic differentiation (Jang, 1993)
Random forest	Bagged CART ensembles by classical optimization (Breiman, 2001)
Decision tree	A single CART tree, when interpretability beats variance reduction
Kernel and boosted methods	Planned; not shipped

Table 1 | What Crucible fits. The neuro-fuzzy family is the interesting one: it's the same inference machinery the analytics extension exposes with hand-authored membership functions, with the parameters learned instead of stated. Same mathematics, different epistemics, different extension.

as the same object can't answer the only question an auditor will ask, which is which of the two it was.

Everything Crucible fits implements one narrow contract — train, predict, predict with probabilities, report the number of classes, save, load — and inference returns through a generic Suggestor that reads typed feature payloads and emits typed prediction payloads with extension provenance. The model sits behind a contract; the Formation sees a Suggestor like any other.

3. The artifact is a fact

The first requirement is that a trained model be representable in the context with real provenance.

> **Definition (Model fact).** A model fact carries the artifact's content hash, the dataset identity and hash, the feature specification, the hyperparameters, the random seed, the library and version that produced it, and the evaluation result on a held-out split. Predictions made by the model carry a reference to it.

With that in place the provenance chain from a decision back to a training run is unbroken: a promoted fact references the prediction, the prediction references the model fact, and the model fact references the dataset and the seed. *Which model decided this, trained on what, and how did it score?* is answerable by walking the ledger rather than by asking whoever ran the notebook.

Reproducibility is what makes the seed meaningful rather than decorative. My forest implementation is deterministic under a fixed seed — same data, same hyperparameters, same seed, same trees — and the artifact serializes to a stable binary format. This matters more in a governed setting than in a research one. A model fact whose artifact can't be reproduced from its stated inputs is an assertion about a file, not a description of a procedure, and when someone asks whether the model that made a decision in March is the model in the registry today, only reproducibility answers.

> **Proposition (Conditional determinism).** Let s_θ be a Suggestor backed by artifact θ. If the model fact for θ is present in the context and pinned in the fleet version, then s_θ satisfies the substrate's determinism axiom over that context. Absent the pin, the axiom holds only per-run, and the substrate's uniqueness result loses its hypothesis.

That's the precise sense in which fitted models are second-class, and the precise price of admission. They are allowed in; they must bring their pin.

4. Training as a Formation

The second requirement concerns the pipeline, and here I am describing a design I have partly built.

The stages of a training run are already Suggestor-shaped. Each reads what it needs from a shared context and proposes something: a dataset agent proposes an ingested dataset; a validation agent proposes a verdict on it; feature engineering proposes a specification; hyperparameter search proposes a configuration; training proposes an artifact; evaluation proposes a score; registry proposes a version; monitoring proposes drift observations; deployment proposes a rollout; a sample-inference agent proposes a smoke test. They exist today as exactly this set of agents, and they run from a command line.

Making them a Formation isn't repackaging. It changes three things.

Order stops mattering. A training script has a sequence, and the sequence encodes assumptions — that validation precedes feature engineering, that evaluation follows training. In a Formation the stages read from context and propose to it, and the engine runs to a fixed point, so a late-arriving validation finding re-enables the stages that depend on it without anyone writing a retry.

Failure stops being terminal. A script that fails at hyperparameter search fails. A Formation whose search Suggestor proposes nothing still has a fixed point — the one reached by the remaining Suggestors — and leaves through one of the substrate's honest exits with a partial result and a reason, rather than a stack trace.

Retraining becomes convergent. This is the part I find most interesting and have least evidence for. A retrain triggered by drift is a new run over a context that already contains the previous model fact, the previous evaluation, and the drift observation that triggered it. Monotonicity means none of that's overwritten: the new artifact is a new fact, and the registry proposal is a proposal about which artifact is current. The history of a model becomes a chain of facts rather than a directory of timestamped files.

> **Open direction (Convergent retraining).** Characterize the conditions under which a retraining Formation reaches a fixed point. The obvious hazard is a cycle: drift triggers retraining, retraining changes predictions, changed predictions produce new outcomes, and new outcomes register as drift. Establish whether a well-founded measure exists, or whether — as I suspect — the budget is the only honest bound.

What exists, plainly: the agents exist, the pipeline runs, and the end-to-end path from a synthetic dataset through a trained artifact to a typed prediction inside a real engine is covered by integration tests. The Formation lift is designed and waits for a

production retrain trigger to justify it. Everything in this section beyond that sentence is an argument, not a report.

5. What the fixture shows

The worked example is deliberately unglamorous: a synthetic loan-default dataset of a thousand samples, a fifty-tree forest, a validation accuracy and a confusion matrix printed by a training binary, and an artifact written to disk. Integration tests then carry that artifact through a real engine to a typed prediction.

The value of the fixture isn't the accuracy, which is a property of synthetic data and means nothing. It's that the whole path is exercised without a proprietary dataset, so the claim *a fitted model can participate in a governed decision with full provenance* is testable by anyone who clones the repository. A demonstration that requires the reader's data to reproduce demonstrates nothing.

6. Limitations

The registry is a proposal surface, not a governance system. It records which artifact is current and how it was produced; it doesn't implement approval workflows, staged rollout, or automatic rollback on degradation. Those are product concerns under my boundary, and naming them as such isn't the same as having solved them.

Monitoring proposes drift observations and doesn't define drift. The threshold separating ordinary variation from a distribution that has moved is supplied, and choosing it badly produces either a retraining loop or a silent staleness. I have no principled default to offer.

Evaluation is a held-out split. It isn't a temporal split, it isn't a subgroup analysis, and it doesn't report calibration. For a governed system this is a real gap: a model fact carrying a single accuracy number carries the least informative summary available (Hastie et al., 2009), and the disaggregated evaluation a consequential decision needs isn't in the artifact.

Finally, the argument of Section 3 is about provenance, not about whether fitting was the right choice. The preceding paper's position stands: where a closed-form method answers the question it's preferable on substrate grounds, and the interpretability literature would add that it's preferable on accountability grounds (Rudin, 2019). Crucible exists for the cases where that's not available, and the temptation it must resist is being used where it is.

7. Conclusion

A fitted model can be a legitimate participant in a governed decision, and what it takes is unglamorous: the artifact must be a fact with a hash, a dataset, a seed and an evaluation; predictions must reference it; and the whole thing must be reproducible from its stated inputs or it's a file rather than a procedure. That's achievable, and it's done.

The larger claim — that a training pipeline should be a Formation converging to a fixed point rather than a script executing in order — is a design I believe in and haven't yet had to prove. Its attraction is that retraining stops being an event that overwrites a model and becomes a run that extends a history. Its hazard is a feedback cycle with no obvious well-founded measure, which is exactly the shape of problem this substrate is meant to make visible rather than to solve.

I'd rather publish the line between what is built and what is argued than blur it. The crucible refines ore; it doesn't pretend the ore was already metal.

> **Code and correspondence.** The work described here is implemented, not sketched. The source behind every claim in this paper — the engine, the extension, the fixtures and the tests that hold the invariants — can be shared on request, including the parts that did not work. Write to kenneth.pernyer@reflective.se. We are equally interested in being told where the argument is wrong.

References

Breiman, L., 2001. Random forests. Machine Learning 45, 5–32.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning, 2nd ed. Springer.

Jang, J.-S.R., 1993. ANFIS: Adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man, and Cybernetics 23, 665–685.

Pernyér, K., 2025a. Nothing to fit: closed-form analytics as governed Suggestors. Reflective Labs technical report.

Pernyér, K., 2025b. Proposal, gate, commitment: deterministic convergence as a substrate for organizational decisions. Reflective Labs technical report.

Pernyér, K., 2025c. Degrees that are not probabilities: fuzzy inference for organizational vagueness. Reflective Labs technical report.

Rudin, C., 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence 1, 206–215.

A. The training agents

T1 Dataset Ingests CSV, TSV or Parquet, with Excel behind a feature flag, and proposes a dataset fact with a content hash.
T2 Validation Proposes a verdict on the dataset — schema, ranges, class balance, missingness. A failing verdict blocks downstream stages rather than annotating them.
T3 Feature engineering Proposes a feature specification. The specification, not the code that produced it, is what the model fact references.
T4 Hyperparameter search Proposes a configuration and the search budget it consumed.
T5 Training Proposes an artifact with its content hash, seed and producing library version.
T6 Evaluation Proposes held-out metrics and a confusion matrix.
T7 Registry Proposes which artifact is current for a given task.
T8 Monitoring Proposes drift observations against a supplied threshold.
T9 Deployment and smoke test Propose a rollout and a sample inference that exercises the artifact through the ordinary Suggestor path.

B. Why the boundary is drawn at fitting

Let g be an inference procedure and Θ its parameters. The analytics extension hosts $s(K) = g(R_s(K), \Theta)$ with Θ supplied in the request; Crucible hosts $s_\theta(K) = g(R_s(K), \theta)$ with θ loaded from an artifact.

The two differ in exactly one place, and it's the place the substrate's proofs quantify over. In the first case s is a function of context alone, so determinism, idempotency and commutativity follow from the argument given in the preceding paper. In the second, s_θ is a *family* of functions indexed by θ, and the substrate's uniqueness result applies to each member separately. Pinning θ selects a member; without the pin, "the same Suggestor" isn't a well-defined object across time, and a proof about a Formation is a proof about a system that no longer exists.

Note what this argument doesn't say. It doesn't say fitted models are less accurate, less useful or less appropriate. It says they are less *self-identifying*, and that the remedy is bookkeeping rather than avoidance.

