The Proxy Manifold: Recipe-Invariant Downstream Prediction from a Standardized Suite of Losses and Probes

Liz Lemma Future Detective

January 17, 2026

Abstract

Scaling-law work such as FLP and FLP-M predicts downstream performance by using pre-training loss as an intermediate variable, exploiting the empirical stability of loss-to-performance mappings after emergence. However, modern (2026-era) training stacks routinely change the recipe—token/parameter ratio, optimizer schedules, curricula, context length, and data processing—so a single validation loss often ceases to be an invariant predictor across runs. We formalize cross-recipe downstream prediction as a transfer problem from correlated checkpoint logs and propose a standardized proxy suite: 10–20 cheap-to-measure quantities combining domain perplexities (as in FLP-M) with targeted capability probe NLLs (e.g., tool-call formatting, long-context retrieval markers, math scratchpads). We provide algorithms for (i) learning a recipe-invariant predictor from multi-recipe logs, (ii) producing calibrated prediction intervals despite checkpoint autocorrelation, and (iii) flagging 'out-of-family' recipes via conformal residual tests. We prove upper bounds on cross-recipe prediction error under a proxy-invariance assumption and give matching lower bounds showing that scalar validation loss alone cannot be uniformly invariant. Experiments (to be included) would validate predictive transfer across recipe variations and demonstrate that the proxy suite improves early decision-making and mixture/recipe search compared to loss-only baselines, while the released harness standardizes proxy measurement for future scaling-law research.

Table of Contents

ciples.

3. 3. Formal Problem Setup: recipes, checkpoints, correlated observations, proxy suite Π, tasks $\mathcal{T}$, and success metrics (error + calibration + OOD detection).

4. 4. Proxy Suite Design (Specification): domain losses, probe NLL construction, measurement protocols, normalization, and cost model; ablations planned.

5. 5. Learning Recipe-Invariant Predictors: model classes (monotone GAM / small monotone MLP / linear), invariance regularization, and multi-task structure.

6. 6. Uncertainty and OOD Detection: block-conformal intervals for correlated checkpoints; conformal residual tests for 'out-of-family' recipe detection; guarantees.

7. 7. Theory: upper bounds under proxy invariance + drift; lower bounds for scalar-loss-only predictors; (optional) proxy selection hardness and approximations.

8. 8. Experimental Protocol (Planned/Optional but Strengthening): multi-recipe small-scale runs; cross-recipe transfer evaluation; early stopping decision value; comparison to FLP/FLP-M baselines.

9. 9. Limitations and Extensions: extending proxies to multimodal/tool traces; interactions with data mixtures; failure modes and governance implications.

10. 10. Artifact Release: proxy harness, reference datasets for probes, evaluation scripts, recommended reporting standard.

1 Introduction and Motivation

Forecasting learning progress (FLP) seeks to infer downstream capabilities from quantities that are cheap to measure during training. In its simplest form, FLP predicts a benchmark metric from a scalar training or validation loss recorded at a checkpoint. This scalar-loss paradigm is attractive because it is universal, already present in most training logs, and strongly correlated with many aggregate performance measures when the training procedure is held fixed. However, our setting is explicitly *multi-recipe*: we consider a family of training recipes $r \in \mathcal{R}$ that differ in optimizer and schedule, token-to-parameter regime, curriculum ordering, context length, and similar interventions. In this regime, scalar-loss FLP becomes brittle: the mapping from a single loss number to a particular downstream task score can change materially under moderate recipe drift, even when the overall loss trajectory appears unchanged.

The source of brittleness is structural. A single loss aggregates heterogeneous error modes that are differently emphasized by different recipes. For instance, shifting the data mixture or curriculum can preserve average loss while reallocating modeling capacity between domains; modifying context length can alter the difficulty distribution of next-token prediction without commensurately shifting short-context validation perplexity; and changes in optimization (e.g., momentum, clipping, warmup) can affect the time at which certain capabilities become reliable, without producing a proportional change in the global loss. In other words, scalar loss fails to identify which aspects of competence have improved. Consequently, two recipes r_1, r_2 may yield checkpoints with comparable scalar validation loss while exhibiting systematically different performance on a task t that depends on a specific latent factor (e.g., instruction-following format adherence, tool-call syntax, retrieval marker usage). This phenomenon is not merely empirical; it is consistent with an information-theoretic obstruction: if the scalar proxy does not encode the latent factor relevant to t, then no predictor depending only on that scalar can be uniformly accurate across recipes.

Our goal is therefore to replace loss-only FLP with a proxy-based predictor that is both informative and robust across the family $\mathcal{R}$. We assume that for each training run i of recipe r and checkpoint s, we can compute a proxy vector $Z_{i,s} \in \mathbb{R}^K$ that concatenates curated domain validation losses L_d and targeted probe negative log-likelihoods Q_j. The design choice is deliberate: the domain losses partially decompose global perplexity into semantically meaningful slices (e.g., code, math, dialogue), while the probes are crafted to detect narrow but operationally important competencies (e.g., structured tool-call formatting, presence of retrieval delimiters). Each coordinate is cheap relative to full benchmark evaluation, yet together the coordinates provide a representation in which recipe-induced variation is more likely to act through Z rather than through an unobserved confounder.

We formalize this desideratum as an invariance hypothesis. Let $Y_{i,s,t}$ denote downstream performance on task $t \in \mathcal{T}$ at checkpoint (i, s), noting that such evaluations are typically sparse. Hypothesis H1 (proxy invariance) posits that there exists a function $f^\star = (f_t^\star)_{t\in\mathcal{T}}$ such that for all recipes $r \in \mathcal{R}$,

$$\mathbb{E}[Y_t \mid Z = z, R = r] = f_t^\star(z),$$

with sub-Gaussian residual noise. The role of H1 is not to claim that recipes are irrelevant, but that recipe dependence is mediated by the proxy vector: once we condition on Z, the conditional expectation of task performance is stable across recipes in-family. This is precisely the kind of assumption under which cross-recipe transfer should be possible from logs, and it also provides a falsifiable target: when H1 fails, we should be able to detect that failure from atypical proxy distributions or residual patterns.

The requirements we impose on an FLP system in this multi-recipe setting are threefold. First, *transfer*: given historical logs $\mathcal{D} = \{(r_i, i, s, Z_{i,s}, Y_{i,s,\cdot})\}$, we seek a predictor $\hat{f}$ that produces accurate predictions for a new recipe r_{new} using only the proxies $Z_{\text{new},s}$ at selected checkpoints. Second, *uncertainty*: because downstream evaluations are sparse and checkpoints within a run are correlated, point predictions alone are insufficient; we require prediction intervals with finite-sample coverage guarantees under appropriate exchangeability assumptions at the run level. Third, *shift detection*: we require a statistically controlled "out-of-family" flag $\text{OOD}(Z, r_{\text{new}})$ that identifies when a new recipe violates the conditions under which the predictor was calibrated, so that a user can fall back to direct evaluation or retraining of the predictor.

A central technical complication is that training logs provide many checkpoints per run, but these checkpoints are strongly dependent. Treating (i, s) as i.i.d. samples leads to overly optimistic error bars and miscalibrated intervals. We therefore treat runs (or blocks of checkpoints within runs) as the primary exchangeable units, introducing an effective sample size N_{eff} governed by a checkpoint correlation time τ. This viewpoint aligns with how recipe variation is instantiated in practice: independent seeds and hyperparameter instantiations produce approximately independent runs, while within-run trajectories are smooth and highly correlated.

Within this framework, our contributions are as follows.

1. We formulate cross-recipe FLP as supervised prediction from proxy vectors Z to downstream task performance Y, with explicit accounting for correlated checkpoints and sparse labels. The formulation makes clear what is observable (proxies at all checkpoints) and what is expensive (downstream tasks evaluated sparsely), and it identifies the invariance condition required for transfer across recipes.

2. We propose a standardized proxy suite Π of moderate dimension ($K \in$

[10, 20]) combining domain validation losses and probe NLLs, together with a learning procedure that encourages recipe-invariant residual structure when fitting $\hat{f}$. The intent is not to fit a separate model per recipe, but to learn a pooled mapping that remains stable across the in-family set.

3. We provide statistically valid uncertainty quantification via block-conformal calibration, yielding prediction intervals $[\ell_t(Z), u_t(Z)]$ with marginal coverage guarantees under run-level exchangeability. The block structure is essential for calibration validity in the presence of within-run dependence.

4. We incorporate an OOD mechanism based on both proxy-distribution shift and residual atypicality, producing a principled "out-of-family" decision rule designed to control false alarms while detecting meaningful violations of H1.

5. We justify the necessity of moving beyond scalar loss by exhibiting a lower bound: there exist recipe pairs for which a scalar validation loss has identical distribution but downstream performance differs by a nontrivial gap, implying that any scalar-loss-only predictor must incur substantial worst-case error on at least one recipe.

The net effect is a shift in perspective: rather than treating loss as a sufficient statistic for capability, we treat a small, standardized proxy vector as the interface between training dynamics and downstream evaluation. Under H1, this interface supports provable cross-recipe generalization and calibrated uncertainty; when H1 fails, it supports controlled detection of that failure. This is the minimal structure we require to make FLP actionable in the regime where recipes drift, scaling regimes change, and downstream benchmarks are too expensive to run exhaustively at every checkpoint.

2 Background and Related Work

Forecasting learning progress (FLP) is the general problem of predicting downstream performance from signals that are available during training, typically as a function of training compute, dataset size, or intermediate losses. Classical instances include learning-curve extrapolation, where one fits parametric or semi-parametric forms to partial trajectories and predicts the eventual value at larger compute. In the modern large-model regime, FLP often takes the form of mapping a checkpoint summary (e.g., training loss, validation loss, or perplexity at a fixed evaluation set) to one or more downstream metrics. A closely related line, sometimes termed FLP-M, addresses *multi-metric* forecasting: we seek to predict a vector of downstream metrics jointly, either to exploit shared structure across tasks or to

provide a consistent early-stopping and model-selection signal across heterogeneous benchmarks. These settings motivate multi-output predictors, as well as methods that share representations or impose structured regularization across tasks. Our work can be viewed as a multi-recipe, multi-metric extension in which the key question is not only within-recipe extrapolation, but stability of the proxy-to-metric mapping under moderate changes to the training procedure.

Scaling laws supply a complementary perspective. Empirical scaling laws posit that certain losses (and sometimes benchmark metrics) follow regular functional relationships with compute, parameters, and data. The canonical results establish approximate power-law behavior for cross-entropy loss as a function of compute and model size over wide ranges, and more recent work refines these relations with compute-optimal tradeoffs between tokens and parameters. Such laws are practically useful for budgeting and for extrapolating average performance, but they are less directly suited to our setting for two reasons. First, scaling laws are typically formulated for a *fixed* training distribution and protocol; changing optimizer details, curricula, context length, or data mixture can shift both the constants and, in some cases, the observed scaling exponents. Second, the quantity that is most robustly power-law (the next-token loss) is precisely the scalar summary that we argue is insufficiently identifying across recipes for particular downstream tasks. Our approach may be interpreted as retaining the spirit of scaling—leveraging cheap-to-measure signals to predict expensive evaluations—while replacing the scalar loss with a structured proxy suite intended to be stable under a family of recipe interventions.

A recurring empirical phenomenon in capability evaluation is so-called *emergence*: certain benchmark scores remain near chance until a threshold scale or training stage, after which they increase rapidly. Whether emergence reflects a genuine phase transition or a smooth curve composed with a sharp metric is debated; nevertheless, it creates a practical difficulty for FLP systems based only on global loss. In particular, a small improvement in loss may coincide with a large jump in a brittle downstream score (e.g., exact-match tasks with strict formatting), and the location of this transition can move with recipe changes (e.g., altered context length or curriculum ordering). As a result, the monotone relationship between loss and performance that holds within a narrow training protocol may fail to transfer across protocols. This motivates proxies that more directly track the onset of the relevant competence (for instance, probe NLLs for format adherence), and it motivates predictor classes that can represent nonlinearity and threshold-like behavior without overfitting to recipe-specific idiosyncrasies.

The effect of data mixing and curricula on downstream behavior is also well documented. Training on mixtures can preserve aggregate perplexity while reallocating capacity across domains; conversely, modest mixture changes can produce large relative changes on specialized tasks even when

overall loss is similar. Curriculum order can matter even when the eventual data distribution is unchanged: early emphasis on certain domains may accelerate the acquisition of skills that later generalize, while delaying others. Moreover, adjustments to sequence length and packing can change the effective difficulty distribution seen by the model (e.g., long-range dependencies versus local token prediction) without a commensurate shift in short-context validation perplexity. These observations suggest that the training recipe acts as an *environment* that can change the conditional distribution of downstream metrics given a coarse proxy. Our proxy suite design is aligned with the standard remedy in distribution shift problems: rather than hope that a single aggregate statistic is sufficient, we attempt to measure a small set of orthogonal axes (domain losses and targeted probes) that make the conditional expectation of downstream outcomes closer to invariant.

Probing provides the technical mechanism for introducing such axes. In representation learning, probes are predictive models (often linear) trained to extract attributes from hidden states; in our setting, we use the term more broadly to include *behavioral probes* implemented as small, cheap evaluation sets that yield a negative log-likelihood for a narrowly defined pattern (e.g., tool-call delimiters, JSON well-formedness cues, retrieval markers). The crucial property is cost: probes are orders of magnitude cheaper than full benchmarks and can be computed at every checkpoint. Unlike general-purpose perplexity, probe losses can be targeted to capabilities that are known to be brittle under recipe changes. While any single probe is narrow, a moderate collection can serve as a low-dimensional summary of multiple latent competences, which is precisely what is required for stable cross-recipe prediction.

Uncertainty quantification is equally central. Even if a point predictor performs well on average, we require statistically meaningful intervals that account for sparse labels and correlated checkpoints. Conformal prediction offers a distribution-free framework for constructing prediction intervals with finite-sample coverage under exchangeability. Standard split conformal methods calibrate a base predictor using held-out residuals; conformalized quantile regression and related variants allow heteroskedastic intervals. However, naive application to checkpoint-level samples is invalid when checkpoints within a run are dependent. The relevant adaptation is *block* or *group* conformal prediction, in which exchangeability is assumed at the level of runs (or blocks of consecutive checkpoints), and calibration is performed on aggregated residuals. This aligns with our operational setting: independent runs are the natural units of replication, while checkpoints are highly autocorrelated observations within each unit. In this sense, our interval construction sits at the intersection of conformal prediction and time-series dependence handling, with the simplifying feature that dependence is largely contained within runs.

Finally, our emphasis on *invariance* connects to domain generalization

and causal prediction principles. A common theme in invariant risk minimization and related approaches is that predictors that rely on spurious correlations vary across environments, whereas predictors based on stable mechanisms exhibit approximately invariant conditional relationships. Translating to our setting, recipes play the role of environments, and Hypothesis H1 asserts that conditioning on our proxies yields an invariant conditional mean across these environments. The practical implication is twofold: we can explicitly encourage invariance by penalizing recipe-dependent residual structure during training, and we can test for violations by monitoring residual atypicality or proxy-distribution shift at inference time. Thus, rather than treating recipe drift as an unmodeled nuisance, we frame it as a structured shift against which we can regularize and, when necessary, raise an out-of-family flag.

In summary, prior work provides (i) the empirical motivation that scalar losses and scaling laws are informative but incomplete under intervention, (ii) the methodological tools of probing to measure targeted competencies cheaply, and (iii) the statistical machinery of conformal prediction and invariance principles to obtain calibrated uncertainty and controlled shift detection. Our contribution is to assemble these elements into a unified multi-recipe forecasting framework that is explicitly designed for correlated checkpoints and sparse downstream labels.

3 Formal Problem Setup

We formalize the forecasting problem in terms of a family of training *recipes* and a fixed collection of downstream tasks. A recipe $r \in \mathcal{R}$ specifies all training-time choices that are held fixed within a run, including (non-exhaustively) optimizer and schedule, token/parameter regime, data mixture and curriculum ordering, context length and packing strategy, and any auxiliary losses. For each recipe r we perform multiple independent training runs indexed by i, differing only by sources of randomness and (optionally) small hyperparameter perturbations that we treat as part of the run index. During a run we save *checkpoints* indexed by s (e.g., step, wall-clock, or token count). We write $C_{i,s}$ for the cumulative training compute (or tokens) consumed by checkpoint s in run i; we do not assume that different recipes share identical checkpoint schedules, but we assume $C_{i,s}$ is observed.

At each checkpoint we compute a K-dimensional proxy vector $Z_{i,s} \in \mathbb{R}^K$, derived from a standardized proxy suite Π. Concretely, Π specifies a set of curated validation domains $\Pi_{\text{dom}} = \{d\}$ and a set of capability probes $\Pi_{\text{probe}} = \{j\}$. The corresponding coordinates of $Z_{i,s}$ are

$$Z_{i,s} = \Big[\{L_d(\text{ckpt}(i,s))\}_{d\in\Pi_{\text{dom}}}, \{Q_j(\text{ckpt}(i,s))\}_{j\in\Pi_{\text{probe}}}\Big],$$

where $L_d(\cdot)$ denotes validation loss/perplexity on domain d, and $Q_j(\cdot)$ de-

notes the negative log-likelihood of a narrowly specified behavioral pattern for probe j. We treat the proxy computation as cheap enough to perform at essentially every checkpoint, in contrast to downstream evaluations.

Downstream evaluations are defined by a finite set of tasks/metrics $\mathcal{T}$. For each task $t \in \mathcal{T}$, checkpoint (i, s) has an associated (possibly unobserved) scalar performance $Y_{i,s,t} \in \mathbb{R}$. The value $Y_{i,s,t}$ may represent accuracy, exact match, pass@k, or any other scalar metric. In our operational regime, the tensor $\{Y_{i,s,t}\}$ is *sparse*: for many checkpoints we only observe proxies, while we only evaluate a subset of (i, s, t) triples due to cost. We denote the resulting log dataset by

$$\mathcal{D} = \{(r_i, i, s, Z_{i,s}, \{Y_{i,s,t}\}_{t \in \mathcal{T}_{i,s}})\},$$

where $\mathcal{T}_{i,s} \subseteq \mathcal{T}$ encodes which tasks were evaluated at checkpoint (i, s). At inference time we are given a new recipe r_{new} and observe proxies $Z_{\text{new},s}$ at one or more checkpoints; our goal is to predict downstream performance (and uncertainty) without executing the full task suite.

A central complication is dependence across checkpoints within the same run. For fixed (r, i), the sequence $\{(Z_{i,s}, Y_{i,s,\cdot})\}_s$ is highly autocorrelated, and treating checkpoints as i.i.d. samples leads to miscalibrated uncertainty and overly optimistic error estimates. We therefore separate two levels of sampling: runs are treated as the primary exchangeable units, while checkpoints within a run are treated as dependent observations. We encode this dependence by an effective correlation time (or block length) τ, which we use both conceptually and algorithmically: statistics computed on checkpoint-level residuals are aggregated into blocks of length τ to obtain an effective sample size N_{eff} that scales with the number of independent runs rather than the number of checkpoints. We will consistently enforce train/calibration splits at the run level, so that no single run contributes to both fitting and calibration.

Our modeling target is a multi-task predictor $f = (f_t)_{t \in \mathcal{T}}$ mapping proxy vectors to downstream performance. We fix a hypothesis class $\mathcal{F}$ (e.g., linear models, generalized additive models, or modest-capacity neural predictors) and seek $\hat{f} \in \mathcal{F}$ fit on $\mathcal{D}$. The formal invariance hypothesis underlying cross-recipe transfer is that conditioning on Z removes recipe dependence in the conditional mean:

$$\text{(H1)} \qquad \mathbb{E}[Y_t \mid Z = z, R = r] = f_t^\star(z) \quad \text{for all } t \in \mathcal{T},\ r \in \mathcal{R}.$$

Equivalently, there exists $f^\star \in \mathcal{F}$ such that the residual $Y_t - f_t^\star(Z)$ has mean zero in every recipe environment. When H1 holds only approximately, we model deviations by bounded drift terms $\Delta_{t,r}(z)$, which will manifest as irreducible cross-recipe error and, operationally, as conditions under which we may wish to flag r_{new} as out-of-family.

We define success through three coupled criteria: predictive accuracy, calibrated uncertainty, and controlled out-of-family detection. First, for point prediction we measure per-task error on held-out runs and (when available) across recipes. A canonical choice is mean squared error

$$\mathrm{MSE}_t(\hat{f}) = \mathbb{E}\big[(\hat{f}_t(Z) - Y_t)^2\big],$$

with expectations taken over checkpoints drawn from new runs of a recipe of interest; we may also report mean absolute error or task-specific proper scoring rules when Y_t is itself a probability-like quantity. Because tasks may have heterogeneous scales, we will consider normalized variants (e.g., z-scored per task) when aggregating across t.

Second, we require prediction intervals $[\ell_t(Z), u_t(Z)]$ such that, for a target miscoverage level α,

$$\mathbb{P}\big(Y_t \in [\ell_t(Z), u_t(Z)]\big) \geq 1 - \alpha,$$

where the probability is with respect to new runs drawn from the in-family distribution. The salient constraint is that the coverage statement must remain valid under within-run dependence, hence our use of run-wise (or block-wise) conformal calibration rather than checkpoint-wise calibration. Interval quality is further assessed by average width and by conditional diagnostics (e.g., coverage stratified by compute C or by proxy regimes).

Third, we require a binary out-of-family decision rule $\mathrm{OOD}(Z, r_{\mathrm{new}})$ indicating whether r_{new} violates the learned proxy-to-metric relationship. We treat this as a hypothesis testing problem with two error modes: false alarms on in-family recipes and missed detections on genuinely shifted recipes. Operationally, we will implement OOD tests using (i) proxy-distribution shift, i.e., whether the observed $Z_{\mathrm{new},s}$ are atypical relative to calibration runs, and (ii) residual atypicality, i.e., whether realized downstream evaluations (when a small number are performed for auditing) produce residuals inconsistent with the calibrated residual distribution. We report standard detection summaries (e.g., false positive rate at a fixed true positive rate) while maintaining an interpretable threshold tied to α and δ-level confidence parameters.

This setup isolates the design degrees of freedom that remain: the choice of proxy suite Π (which determines K and the semantic axes encoded in Z), the model class $\mathcal{F}$ and any invariance-promoting regularization used to fit $\hat{f}$, and the measurement protocol by which Z is computed consistently across recipes and checkpoints. We turn next to the concrete specification of Π, including domain and probe construction, normalization, and a cost model suitable for routine checkpoint-level evaluation.

4 Proxy Suite Design (Specification)

We now specify the proxy suite Π that induces the checkpoint-level vector $Z \in \mathbb{R}^K$. The design goal is twofold: (i) to capture the principal axes

along which downstream behavior varies across training compute and across recipes, and (ii) to ensure that proxy measurement is sufficiently standardized that differences in Z reflect model state rather than evaluation artifacts. Throughout we target $K \in [10, 20]$ so that proxy evaluation remains routine at essentially every checkpoint.

Domain validation losses. The first component of Π is a set of curated validation domains $\Pi_{\text{dom}} = \{d\}$. Each domain d is a fixed corpus slice (or mixture) with a frozen tokenization and sequence construction rule. For a checkpointed model θ, we define the domain proxy by teacher-forced cross-entropy on that slice:

$$L_d(\theta) = \frac{1}{|\mathcal{S}_d|} \sum_{x \in \mathcal{S}_d} \Big(-\frac{1}{|x|} \sum_{u=1}^{|x|} \log p_\theta(x_u \mid x_{<u}) \Big),$$

where $\mathcal{S}_d$ is the fixed evaluation set for domain d, and sequences are constructed with a deterministic packing rule (defined below). We treat L_d as a generic negative log-likelihood proxy; whether one reports loss, bits-per-byte, or perplexity is immaterial provided we fix a monotone transform. In practice we choose domains to be semantically distinct and operationally relevant (e.g., general web text, code, mathematics, instruction-style dialogs, multilingual text), since a single aggregate loss can obscure meaningful trade-offs. The suite is *not* intended to approximate the full pretraining mixture, but rather to provide a low-dimensional coordinate system for cross-recipe comparison.

To reduce variance and improve comparability across checkpoints, each $\mathcal{S}_d$ is held constant over the entire study (no refresh), sized so that the estimator variance of L_d is negligible compared to inter-checkpoint changes at the correlation scale τ. We additionally stratify $\mathcal{S}_d$ by sequence length buckets and report length-weighted averages, which prevents spurious drift when a recipe changes context length but the model's token-level competence is similar.

Capability probes via NLL. The second component $\Pi_{\text{probe}} = \{j\}$ consists of narrowly specified probes designed to capture discrete behaviors that domain losses alone often underidentify. A probe j is a distribution over prompts x together with a target completion y (possibly structured), and its score is the conditional NLL under teacher forcing:

$$Q_j(\theta) = \frac{1}{|\mathcal{P}_j|} \sum_{(x,y) \in \mathcal{P}_j} \Big(-\frac{1}{|y|} \sum_{u=1}^{|y|} \log p_\theta(y_u \mid x, y_{<u}) \Big).$$

By construction, Q_j depends only on the model's conditional distribution and is independent of decoding heuristics. Probes can target, for example: (i)

adherence to tool-call or API schemas (well-formed JSON, function signature arguments), (ii) instruction-following markers (e.g., refusal formats, delimiter usage), (iii) retrieval or citation markers (presence and correct placement of `[ref]` tokens), and (iv) syntactic fidelity in code (balanced brackets, import lines). We emphasize that probes are *not* downstream benchmarks; they are short, cheap evaluations whose purpose is to detect capability emergence or regressions that may not be visible in broad-domain losses.

To avoid trivial leakage, probe items are synthetically generated from templates with held-out parameter seeds, and the entire probe set is fixed prior to analyzing recipes. Each probe is also constructed to be robust to superficial formatting: we either canonicalize whitespace and compare token sequences under the canonicalization, or we specify target strings in a tokenizer-aligned manner. For probes with multiple acceptable outputs, we compute a $\log \sum \exp$ likelihood over acceptable completions (or, operationally, the minimum NLL among a small finite set of canonical variants), thereby reducing sensitivity to arbitrary stylistic choices.

Measurement protocol and standardization. Proxy comparability requires that L_d and Q_j be measured under a single evaluation harness. We therefore fix: (i) tokenizer version and vocabulary, (ii) context window W_{eval} used for evaluation (truncating or segmenting longer sequences deterministically), (iii) packing strategy (e.g., pack to W_{eval} without crossing document boundaries), and (iv) numerical precision and attention masking conventions. When a recipe trains with a different context length W_{train}, we still evaluate at the fixed W_{eval} to ensure that Z reflects learned conditional distributions rather than changes in evaluation length. We also fix dropout and stochastic layers to evaluation mode. For reproducibility, we store the exact evaluation manifests (file hashes, sampling seeds, and template seeds) and log the software commit that produced each proxy vector.

Because checkpoints within a run are correlated, proxy measurement noise should be small enough that observed variation in $Z_{i,s}$ is dominated by genuine training progress rather than Monte Carlo error. We thus choose evaluation set sizes so that the standard error of each coordinate is well below the typical between-block change over τ steps. When this is not feasible for a coordinate (e.g., expensive long-context probes), we explicitly log an uncertainty estimate and treat that coordinate as noisy in later modeling.

Normalization and coordinate transformations. Raw losses across domains and probes have heterogeneous scales and, in some cases, nonstationary variance across training compute. We therefore define a normalization map $\phi : \mathbb{R}^K \to \mathbb{R}^K$ applied before fitting predictors. The default is per-coordinate affine normalization using statistics computed on the training

split at the run level:

$$\tilde{Z}_k = \phi_k(Z_k) = \frac{Z_k - \mu_k}{\sigma_k}, \qquad (\mu_k, \sigma_k) \text{ estimated on training runs only.}$$

For heavy-tailed coordinates (often probes early in training), we optionally use robust scale estimates (median and MAD) or a monotone transform such as $\log(Z_k)$ when $Z_k > 0$ and multiplicative changes are more stable than additive ones. We avoid checkpoint-wise normalization schemes that could leak information across time within a run. If monotonicity constraints are later imposed on select coordinates, we apply only monotone transforms so that the ordering information is preserved.

Cost model. Let T_d denote the number of evaluation tokens in domain slice d and T_j the number of completion tokens in probe j. Proxy computation at a checkpoint is essentially a teacher-forced forward pass over $\sum_d T_d + \sum_j T_j$ tokens. Hence the marginal cost of proxies is proportional to token throughput and scales linearly in K for fixed per-coordinate token budgets:

$$\text{cost}(\Pi) \approx c_{\text{fw}} \Big(\sum_{d \in \Pi_{\text{dom}}} T_d + \sum_{j \in \Pi_{\text{probe}}} T_j \Big),$$

where c_{fw} is the per-token forward cost at the evaluation precision. We choose T_d, T_j so that $\text{cost}(\Pi)$ is negligible relative to training cost at checkpoint cadence (e.g., $\ll 1\%$ of the compute between checkpoints). This constraint effectively upper-bounds both the number of domains and the complexity of probes, motivating short, template-based probes rather than full interactive tasks.

Planned ablations. To validate that Π is neither overfit nor redundant, we pre-specify ablations along five axes. (i) *Scalar baseline*: replace Z by a single aggregate validation loss (or perplexity) to empirically instantiate the failure mode of scalar proxies. (ii) *Domain-only* versus *probe-only*: remove one component and measure cross-recipe degradation, testing whether probes capture complementary information. (iii) *Suite size*: vary K by subsampling coordinates and measure the prediction–cost frontier. (iv) *Normalization sensitivity*: compare affine, robust, and log-normalized variants of ϕ, verifying that conclusions are not artifacts of scaling. (v) *Protocol stress tests*: intentionally perturb evaluation settings (packing, W_{eval}, precision) to quantify how much proxy drift can be induced by measurement choices, thereby bounding the degree of standardization required for stable transfer. These ablations are evaluated at the run level to respect dependence, and the results determine the minimal proxy suite that supports the subsequent recipe-invariant learning procedure.

5 Learning Recipe-Invariant Predictors

We now describe how we learn a predictor $\hat{f} \in \mathcal{F}$ mapping proxies to downstream performance in a manner that transfers across recipes. For each task $t \in \mathcal{T}$, our target is a function $f_t : \mathbb{R}^K \to \mathbb{R}$ such that $f_t(Z)$ approximates $\mathbb{E}[Y_t \mid Z]$ and, under Hypothesis H1, this conditional expectation does not depend on the recipe R. We work with normalized proxies $\tilde{Z} = \phi(Z)$ as defined previously, and we treat downstream labels as sparse: for many (i, s) we observe $\tilde{Z}_{i,s}$ but only a subset of $\{Y_{i,s,t}\}_{t \in \mathcal{T}}$.

Multi-task prediction with missing labels. Let $\mathcal{I}$ index the set of checkpoint evaluations available for training (after any block aggregation used to reduce within-run dependence). For each $m \in \mathcal{I}$ we have $(\tilde{Z}_m, R_m)$, and for each task t we may or may not have an observed label $Y_{m,t}$. We introduce a mask $M_{m,t} \in \{0, 1\}$ indicating whether $Y_{m,t}$ is observed. Our fitting objective is therefore a masked empirical risk of the form

$$\min_{f \in \mathcal{F}} \sum_{m \in \mathcal{I}} \sum_{t \in \mathcal{T}} M_{m,t}\, \ell\Big(f_t(\tilde{Z}_m),\, Y_{m,t}\Big) \;+\; \Omega(f) \;+\; \lambda\, \mathcal{R}_{\text{inv}}(f),$$

where ℓ is typically squared loss for real-valued metrics (or a proper scoring loss for log-odds-type targets), Ω is a standard capacity control term (e.g., ridge weight decay or spline smoothness), and $\mathcal{R}_{\text{inv}}$ encourages recipe-invariance of residual structure. The mask-based formulation allows us to train a single multi-task model even when tasks are evaluated at different cadences.

Invariance regularization across recipes. Under H1, for each t the residual $\varepsilon_{m,t} := Y_{m,t} - f_t(\tilde{Z}_m)$ should have mean 0 conditional on $\tilde{Z}_m$ and should not exhibit systematic recipe dependence. We operationalize this as a penalty on recipe-wise residual means. Let $\bar{\varepsilon}_{r,t}(f)$ denote the average residual for task t on points from recipe r, computed on the training split:

$$\bar{\varepsilon}_{r,t}(f) := \frac{\sum_{m \in \mathcal{I}} \mathbf{1}\{R_m = r\}\, M_{m,t}\, \big(Y_{m,t} - f_t(\tilde{Z}_m)\big)}{\sum_{m \in \mathcal{I}} \mathbf{1}\{R_m = r\}\, M_{m,t}}.$$

We then define

$$\mathcal{R}_{\text{inv}}(f) := \sum_{t \in \mathcal{T}} \mathrm{Var}_{r \in \mathcal{R}_{\text{train}}}\big(\bar{\varepsilon}_{r,t}(f)\big),$$

where $\mathcal{R}_{\text{train}}$ is the set of recipes appearing in the training split and Var is computed with recipe weights proportional to the number of labeled points for that task. This penalty is minimal when residual means align across recipes, which is a weak but empirically useful proxy for recipe-invariant conditional expectations. One may strengthen the notion by penalizing recipe dependence of richer residual summaries, e.g., by matching residual quantiles

across recipes or by fitting a recipe classifier on residuals and adversarially minimizing its accuracy; we restrict attention to the mean-based penalty because it is stable in the sparse-label regime and interacts cleanly with convex model classes.

We choose λ by cross-recipe validation: we hold out entire recipes (or, when recipes are few, hold out entire runs within each recipe) and select λ to minimize worst-case validation error over held-out environments. This selection criterion aligns with our objective of controlling cross-recipe generalization rather than average within-recipe fit.

Model classes. We consider three nested predictor families $\mathcal{F}$, trading off interpretability, monotonic structure, and flexibility.

(i) Linear models. The simplest choice is task-wise linear prediction

$$f_t(\tilde{z}) = w_t^\top \tilde{z} + b_t,$$

fitted with ridge regularization $\Omega(f) = \sum_t \|w_t\|_2^2$. This model is attractive for two reasons: (a) it yields transparent feature attributions in the proxy coordinates, and (b) it supports finite-sample bounds under standard sub-Gaussian assumptions (cf. Thm. 1). In the multi-task setting we optionally impose structured sharing across tasks, e.g., a group-lasso penalty $\sum_{k=1}^{K} \|(w_{t,k})_{t\in\mathcal{T}}\|_2$ to encourage a common subset of informative proxies. Such sharing is beneficial when some tasks have few labels, since it borrows statistical strength through feature selection rather than forcing identical task functions.

(ii) Monotone generalized additive models (GAMs). Linear models can underfit when proxy–metric relationships saturate or exhibit threshold effects. A GAM represents each task as

$$f_t(\tilde{z}) = b_t + \sum_{k=1}^{K} g_{t,k}(\tilde{z}_k),$$

where each $g_{t,k}$ is a univariate smooth function (e.g., cubic spline or piecewise-linear basis) and $\Omega(f)$ penalizes curvature or total variation. Crucially, GAMs admit coordinate-wise monotonicity constraints on selected proxies. Since most proxy coordinates are losses or NLLs, it is often reasonable (in the post-emergence regime) to require that decreasing a loss should not decrease predicted downstream performance. Formally, for a designated monotone set $\mathcal{K}_\uparrow \subseteq [K]$ we enforce $g'_{t,k}(\cdot) \le 0$ (nonincreasing) for $k \in \mathcal{K}_\uparrow$. In practice we implement this by constraining the spline coefficients to yield nonpositive finite differences on a fixed grid. We stress that monotonicity is a modeling choice rather than a theorem: we enable it only for proxies whose measurement protocol is stable and whose relationship to the task is empirically monotone on held-out recipes.

(iii) Small monotone MLPs. When interactions between proxies are essential (e.g., a task improves only when both code loss and tool-format probe NLL cross certain thresholds), additive structure can be too restrictive. We therefore also consider a small neural predictor with a shared trunk $h : \mathbb{R}^K \to \mathbb{R}^m$ and task-specific heads $f_t(\tilde{z}) = a_t^\top h(\tilde{z}) + b_t$. To preserve monotonicity in selected coordinates, we use a constrained architecture: weights from monotone coordinates are parameterized to be nonnegative (via softplus reparameterization), activations are monotone (e.g., ReLU or softplus), and the overall sign is chosen so that lower losses map to higher predicted scores (equivalently, we input $-\tilde{z}_k$ for loss-like coordinates). This yields a coordinate-wise monotone function for the designated subset while allowing unconstrained coordinates (e.g., context-length indicators or compute) to enter freely. Compared to unconstrained MLPs, we find that imposing even partial monotonicity materially reduces cross-recipe brittleness, consistent with the hypothesis that many recipe changes primarily affect the rate at which proxy losses decrease rather than the directionality of their relationship to downstream competence.

Incorporating compute and checkpointing. Although our predictors are defined as functions of Z alone under H1, in finite data it is often useful to include a scalar compute proxy (e.g., $\log C$) as an additional coordinate. This allows the model class to represent residual learning dynamics not captured by the proxy suite (for instance, systematic late-training improvements on a task even after proxies plateau). When compute is included, we treat it as a non-monotone coordinate by default, since different recipes can shift learning speed; invariance regularization then encourages the predictor to rely on compute only insofar as it provides recipe-stable information conditional on the other proxies.

Practical diagnostics. After fitting $\hat{f}$, we compute recipe-wise residual summaries on held-out runs and verify that (i) residual means are near zero and exhibit low between-recipe variance, and (ii) large residuals are not concentrated on a single recipe or proxy region. When these diagnostics fail, we interpret it as evidence either that H1 is violated for the current proxy suite Π or that the model class $\mathcal{F}$ is misspecified; in either case, the subsequent uncertainty and OOD procedures are expected to activate, and we treat the mapping as unreliable for extrapolation to new recipes.

Uncertainty quantification with correlated checkpoints. Our predictor $\hat{f}$ yields a point estimate $\hat{f}_t(\tilde{z})$ for each $t \in \mathcal{T}$, but for decision-making we require calibrated uncertainty that remains valid when checkpoints within a run are temporally correlated. We therefore adopt a split-conformal construction in which the exchangeable units are entire *runs* (or, more finely,

nonoverlapping *blocks* of checkpoints whose length exceeds the correlation time). Concretely, we partition the set of runs into disjoint training and calibration sets, $\mathcal{J}_{\mathrm{tr}} \cup \mathcal{J}_{\mathrm{cal}}$, fit $\hat{f}$ on $\mathcal{J}_{\mathrm{tr}}$ as in §5, and calibrate intervals using only $\mathcal{J}_{\mathrm{cal}}$. This run-wise split is essential: it prevents leakage through highly correlated checkpoints and provides the exchangeability required by the conformal validity argument.

Block aggregation and nonconformity scores. Fix a block length $\tau \in \mathbb{N}$. For each calibration run $i \in \mathcal{J}_{\mathrm{cal}}$, we partition its checkpoints into nonoverlapping blocks $B_{i,1}, \ldots, B_{i,B_i}$, where each block contains τ consecutive checkpoints (except possibly the final remainder, which we drop for simplicity). Within each block, we define a representative proxy $\bar{Z}_{i,b}$; in practice we take either the proxy at the final checkpoint of the block (to preserve a forward-looking interpretation) or the block average $\bar{Z}_{i,b} := \frac{1}{|B_{i,b}|} \sum_{s \in B_{i,b}} \tilde{Z}_{i,s}$, which reduces measurement noise. When a task label is available at some checkpoint in the block, we analogously define a representative label $\bar{Y}_{i,b,t}$ (e.g., the label at the final checkpoint in the block, or an average if multiple evaluations occur within the block). We then compute absolute-residual nonconformity scores

$$E_{i,b,t} \;:=\; \left|\bar{Y}_{i,b,t} - \hat{f}_t(\bar{Z}_{i,b})\right| \qquad \text{for all observed } (i,b,t).$$

When labels are sparse, many triples (i,b,t) are missing; we simply calibrate each task t on the available block-level residuals for that task, and we record the corresponding calibration sample size m_t.

The role of τ is to ensure that $\{E_{i,b,t}\}_b$ behaves approximately as an i.i.d. sequence within each run after aggregation. For theoretical statements it is cleanest to treat each run as contributing at most one exchangeable score, e.g.,

$$S_{i,t} \;:=\; \max_{1 \le b \le B_i} E_{i,b,t},$$

which yields simultaneous validity for all blocks within the run. In practice, using all nonoverlapping blocks typically improves efficiency; we view this as replacing the raw sample size by an effective sample size governed by τ, in the spirit of the N_{eff} abstraction used in our bounds.

Split-conformal intervals (marginal coverage). For each task t, let $\{S_{i,t}\}_{i \in \mathcal{J}_{\mathrm{cal}}}$ denote the calibration scores (one per calibration run, using the within-run maximum as above, and ignoring runs with no labels for task t). Let m_t be the number of available calibration runs for task t, and define the conformal quantile

$$q_t \;:=\; \mathrm{Quantile}_{1-\alpha}\Big(\{S_{i,t}\}_{i \in \mathcal{J}_{\mathrm{cal}}}\Big) \;=\; \text{the } \lceil (m_t+1)(1-\alpha) \rceil \text{-th smallest value.}$$

We then report the symmetric prediction interval

$$[\ell_t(\tilde{z}), u_t(\tilde{z})] \;:=\; \big[\hat{f}_t(\tilde{z}) - q_t,\ \hat{f}_t(\tilde{z}) + q_t\big].$$

Under exchangeability of runs within the in-family distribution and the run-wise split, the standard split-conformal argument applies with runs as the exchangeable units, yielding the marginal coverage guarantee stated in Thm. 3. We emphasize that we do *not* require $\hat{f}$ to be correct; conformal calibration corrects whatever systematic miscalibration remains on the calibration runs, provided the new run is drawn from the same in-family distribution.

When heteroskedasticity is pronounced (e.g., residual scale varies across proxy regions), we optionally use a normalized score $E_{i,b,t}/\hat{s}_t(\bar{Z}_{i,b})$, where $\hat{s}_t$ is a learned scale model fitted on training data, and then multiply the calibrated quantile by $\hat{s}_t(\tilde{z})$ at inference time. This is a standard variance-stabilization step and preserves validity when the score is computed consistently on calibration and test.

Multiple tasks and simultaneous statements. The above intervals are marginally valid per task. If we require a family-wise statement over $\mathcal{T}$ (e.g., all task outcomes lie within their respective intervals with probability at least $1-\alpha$), we may use a Bonferroni correction by calibrating with level $\alpha/|\mathcal{T}|$ for each task, or we may calibrate a single joint score such as

$$S_i^{\text{joint}} \;:=\; \max_{t\in\mathcal{T}} \frac{S_{i,t}}{\hat{q}_t^{(0)}},$$

where $\hat{q}_t^{(0)}$ is a preliminary scale (e.g., the median of $S_{i,t}$), and then inflate all task-wise radii by the calibrated quantile of S_i^{joint}. We use the joint construction when a single conservative uncertainty budget is desired across metrics.

Residual-based out-of-family detection. Intervals alone do not distinguish "hard but in-family" from "mapping failure due to recipe shift." We therefore add an OOD procedure that is explicitly tied to Hypothesis H1. Suppose that for a new recipe r_{new} we can afford a small number of downstream evaluations at selected checkpoints, yielding block-level pairs $(\bar{Z}_{\text{new},b}, \bar{Y}_{\text{new},b,t})$. We compute scores $E_{\text{new},b,t} = |\bar{Y}_{\text{new},b,t} - \hat{f}_t(\bar{Z}_{\text{new},b})|$ and an aggregated run score $S_{\text{new},t} := \max_b E_{\text{new},b,t}$. For each t, we form a conformal p-value

$$p_t \;:=\; \frac{1 + \sum_{i\in\mathcal{J}_{\text{cal}}} \mathbf{1}\{S_{i,t} \geq S_{\text{new},t}\}}{m_t + 1}.$$

Under the in-family exchangeability assumption, p_t is super-uniform, hence thresholding $p_t \leq \gamma$ controls the false-positive rate at level γ for each task. To obtain a single binary flag, we may combine evidence across tasks by

$p_{\min} := \min_{t \in \mathcal{T}_{\text{eval}}} p_t$ on the subset $\mathcal{T}_{\text{eval}}$ of tasks actually evaluated for the new run, and apply a Holm–Bonferroni correction (or, more simply, compare $p_{\min}$ to $\gamma/|\mathcal{T}_{\text{eval}}|$). Intuitively, if H1 holds then residual behavior on the new recipe should be statistically indistinguishable from calibration residual behavior; repeated exceedances indicate that the mapping from proxies to outcomes has drifted beyond what calibration witnessed.

Proxy-distribution shift tests without labels. When downstream labels are unavailable (or too sparse for meaningful residual tests), we still wish to detect gross departures in proxy space. We treat OOD detection as a conformal *membership* test in the proxy distribution: define a nonconformity score $A(z)$ that is large when z is atypical under calibration proxies, for instance

$$A(z) := \min_{m \in \mathcal{I}_{\text{cal}}} \|z - \tilde{Z}_m\|_2 \qquad \text{or} \qquad A(z) := (z - \hat{\mu})^\top \hat{\Sigma}^{-1} (z - \hat{\mu}),$$

where $\mathcal{I}_{\text{cal}}$ indexes calibration blocks and $(\hat{\mu}, \hat{\Sigma})$ are estimated on calibration proxies. For a new run we compute $A(\bar{Z}_{\text{new},b})$ across blocks and aggregate via $A_{\text{new}} := \max_b A(\bar{Z}_{\text{new},b})$. We then compute a p-value by ranking A_{new} among the analogous calibration run scores. Under exchangeability, this p-value is again super-uniform, yielding a controlled false-positive OOD flag. This test detects recipes that induce proxy vectors outside the calibration manifold (e.g., different tokenization, radically different curricula, or context-length regimes that alter probe behavior), even before we spend budget on downstream benchmarks.

Interpretation and failure modes. If the proxy-shift test fires but residual tests cannot be run, we interpret the recipe as out-of-support in proxy space and decline to extrapolate. If proxy-shift does not fire but residual tests do, we interpret this as a direct violation of H1: the proxy suite may be insufficient to capture recipe-dependent changes that matter for downstream tasks (cf. the impossibility phenomenon formalized later for scalar proxies). Finally, if neither fires, we treat the new recipe as in-family and rely on the conformal intervals as our operational uncertainty quantification mechanism.

6 Theory: guarantees and limitations

Block-dependent observations and an effective sample size. We formalize the dependence induced by temporally adjacent checkpoints by assuming that, within a run i, the sequence $\{(Z_{i,s}, Y_{i,s,\cdot})\}_{s \geq 1}$ is τ-dependent or at least strongly mixing with correlation time τ. After block aggregation at length τ, we obtain approximately independent block-level samples $\{(\bar{Z}_{i,b}, \bar{Y}_{i,b,\cdot})\}_{b=1}^{B_i}$ within each run. Let n denote the number of independent

runs and let $B := \sum_{i=1}^{n} B_i$ be the number of blocks contributing labels. We summarize the usable statistical information by an effective sample size N_{eff}, which one may take as $N_{\text{eff}} := B$ in the idealized i.i.d. block model, or more conservatively $N_{\text{eff}} := \sum_{i=1}^{n} \min\{B_i, 1\}$ if we treat each run as contributing a single exchangeable unit. All of our rates are stated in terms of N_{eff}, thereby separating temporal-correlation modeling from the predictor analysis.

Upper bounds under proxy invariance: linear special case. We first consider the regime in which proxy invariance holds exactly and the conditional mean is linear. Fix a task $t \in \mathcal{T}$ and assume

$$Y_t \;=\; w_t^{\star\top} Z + b_t^{\star} + \epsilon_t, \qquad \mathbb{E}[\epsilon_t \mid Z, R] = 0, \qquad \epsilon_t \text{ is } \sigma\text{-sub-Gaussian},$$

with the same $(w_t^{\star}, b_t^{\star})$ for all recipes $r \in \mathcal{R}$. If we fit a pooled ridge estimator $\hat{f}_t(z) = \hat{w}_t^{\top} z + \hat{b}_t$ on block-aggregated training data (with run-wise splitting as above), standard concentration for ridge regression yields a finite-sample error bound of the form

$$\mathbb{E}_{(Z,Y)\sim r_{\text{new}}}\big[(\hat{f}_t(Z) - Y_t)^2\big] \;\le\; c \cdot \frac{\sigma^2 K \log(1/\delta)}{N_{\text{eff}}} \tag{1}$$

with probability at least $1 - \delta$, for a numerical constant $c > 0$ depending on regularization and mild moment conditions on Z. A union bound over $t \in \mathcal{T}$ converts (1) into a simultaneous statement for all tasks by replacing δ with $\delta/|\mathcal{T}|$. In particular, to achieve mean-squared prediction error at most ε^2 (ignoring constants), it suffices that

$$N_{\text{eff}} \;\gtrsim\; \frac{\sigma^2 K \log(|\mathcal{T}|/\delta)}{\varepsilon^2}.$$

The salient point is that, under invariance, cross-recipe transfer is not an additional difficulty beyond the usual K/N_{eff} scaling: once Z is informative and the mapping is shared, we can pool across recipes without bias.

Beyond linearity: complexity control and invariance regularization. For a general predictor class $\mathcal{F}$ (e.g., multi-task MLPs, kernel methods, or monotone additive models), the same structure persists with the parametric factor K replaced by a suitable complexity measure (Rademacher complexity, covering numbers, or stability bounds for the chosen optimizer). Concretely, let $\hat{f}$ be an empirical risk minimizer over $\mathcal{F}$ on training runs, optionally augmented with a recipe-invariance penalty that discourages recipe-dependent mean residuals. Under exact proxy invariance (Hypothesis H1) and standard uniform convergence assumptions, we obtain for each t a decomposition

$$\mathbb{E}\big[(\hat{f}_t(Z) - Y_t)^2\big] \;\le\; \underbrace{\mathbb{E}\big[(f_t^{\star}(Z) - Y_t)^2\big]}_{\text{irreducible noise}} \;+\; \underbrace{O(\text{Comp}(\mathcal{F})/N_{\text{eff}})}_{\text{estimation}} \;+\; \underbrace{\text{OptErr}(\hat{f}; \mathcal{F})}_{\text{optimization/approximation}},$$

where $\mathrm{Comp}(\mathcal{F})$ is an appropriate complexity term. The invariance penalty is not required for validity when H1 is true, but it improves robustness when recipes are heterogeneous in marginal proxy distribution: it suppresses predictors that interpolate spurious recipe identifiers through Z and thereby reduces the variance of cross-recipe extrapolation.

Bounded recipe drift and the bias–variance tradeoff. Exact invariance is an idealization; we therefore quantify controlled violations. Assume that for each recipe r and task t,

$$\mathbb{E}[Y_t \mid Z = z, R = r] \;=\; f_t^\star(z) + \Delta_{t,r}(z), \qquad \sup_z |\Delta_{t,r}(z)| \le \eta.$$

Then any estimator $\hat{f}$ with an in-family estimation error bound (e.g., (1) in the linear case) obeys the bias–variance inequality

$$\mathbb{E}\big[(\hat{f}_t(Z) - Y_t)^2\big] \;\le\; O\!\left(\frac{\sigma^2\,\mathrm{Comp}}{N_{\mathrm{eff}}}\right) \;+\; O(\eta^2),$$

where $\mathrm{Comp} = K\log(1/\delta)$ in the ridge setting. The term η^2 is irreducible from in-family data: it represents recipe-specific systematic deviation not captured by Z. This decomposition clarifies the role of our OOD procedures. Residual-based flags are, in effect, tests for whether η for r_{new} is small enough that conformal calibration remains informative; proxy-shift flags are tests for whether r_{new} lies outside the proxy support on which $f^\star$ was learned. In both cases the theoretical message is the same: when drift is bounded, prediction error degrades gracefully; when drift is unbounded, no method can guarantee accuracy without additional labels.

Lower bound: scalar loss cannot be uniformly recipe-invariant. We next formalize a limitation that motivates a multi-dimensional proxy suite. Consider a latent decomposition $Z = (U, V)$ where U controls average validation loss and V controls a downstream capability (e.g., tool formatting), and assume the scalar proxy S is a function only of U (e.g., $S = L_{\mathrm{avg}}$). We construct two recipes r_1, r_2 such that U has the same distribution under both recipes, hence S is identically distributed, but the distribution of V differs so that $\mathbb{E}[Y_t \mid R = r_1] - \mathbb{E}[Y_t \mid R = r_2] = \Delta$ for some $\Delta > 0$. Then any predictor $g(S)$ cannot distinguish r_1 from r_2 and must incur worst-case absolute error at least $\Delta/2$ on one of them:

$$\max\Big\{\mathbb{E}_{r_1}\big[|g(S) - Y_t|\big],\; \mathbb{E}_{r_2}\big[|g(S) - Y_t|\big]\Big\} \;\ge\; \Delta/2.$$

This impossibility persists even with infinite data, because it is information-theoretic: the scalar proxy discards a degree of freedom that recipes can manipulate without changing loss. The practical consequence is that "validation loss only" early stopping or model selection cannot be uniformly reliable

across curricula or context/token regimes; targeted probes are required to restore identifiability of the mapping.

(Optional) proxy suite selection: hardness and approximations. Finally, we note that choosing *which* proxies to include is itself combinatorial. Given a library of m candidate proxies and a budget K, selecting the subset that minimizes worst-case cross-recipe prediction error is NP-hard even for linear predictors, by reduction from sparse subset selection or Set Cover: recipes/tasks induce constraints that can be satisfied only if particular proxies are included, and deciding whether K proxies suffice encodes the cover decision. Accordingly, we do not seek an optimal suite. Instead, we either (i) fix a standardized suite Π informed by domain knowledge, or (ii) replace the true objective with a submodular surrogate (e.g., mutual information under a Gaussian model), in which case the greedy algorithm achieves a $(1 - 1/e)$-approximation under a cardinality constraint. This justifies the pragmatic stance that proxy design should be stable and auditable, while learning and calibration handle the remaining statistical uncertainty.

Experimental protocol (planned; strengthening evidence). We outline a small-scale but multi-recipe experimental protocol intended to stress-test Hypothesis H1 and the full PROXY-MANIFOLD pipeline (prediction, interval calibration, and OOD flagging) under realistic logging constraints. Our goal is not to maximize absolute task scores, but to measure *cross-recipe predictability* of downstream behavior from cheap proxies, and to quantify the decision value of such predictions for early stopping and evaluation allocation.

Recipes and runs. We instantiate a finite family $\mathcal{R}_{\text{exp}} \subset \mathcal{R}$ of recipes that vary along axes known to induce nontrivial training dynamics: optimizer family and momentum (AdamW vs. Adafactor), learning-rate schedule shape (cosine vs. linear warmup/decay), token/parameter ratio, curriculum order (domain mixing weights over time), context length regime (fixed vs. staged increases), and (optionally) minor architectural knobs that do not obviously change representational capacity (e.g., activation checkpointing or attention implementation details). For each $r \in \mathcal{R}_{\text{exp}}$ we run n_r independent seeds, each producing a checkpoint trajectory $\{Z_{i,s}, Y_{i,s,\cdot}\}_s$ with dense proxy logging and sparse downstream evaluation. We choose the total number of training runs $n = \sum_r n_r$ to be large enough that run-wise splits yield a nontrivial calibration set; in particular, calibration is performed *only* on held-out runs to preserve exchangeability at the run level.

Proxy logging and downstream evaluation schedule. At every checkpoint s (or at a fixed cadence), we compute the proxy vector

$$Z_{i,s} \;=\; \big[\{L_d\}_{d\in\Pi_{\text{dom}}},\; \{Q_j\}_{j\in\Pi_{\text{probe}}}\big] \in \mathbb{R}^K,$$

where Π_{dom} are curated domain validation losses and Π_{probe} are targeted probe NLLs. Downstream task evaluations $Y_{i,s,t}$ are expensive; we therefore evaluate them on a sparse schedule designed to decouple the measurement problem from the training problem. Concretely, for each run we evaluate (i) a small fixed set of anchor checkpoints (early/mid/late), plus (ii) a small number of randomly selected checkpoints, yielding sparse labels that are approximately uniform over training compute C. This schedule supports both point-prediction training and calibrated intervals while reducing the risk that our predictor learns an artifact of a deterministic evaluation cadence.

Cross-recipe transfer and held-out recipe tests. To test whether learned mappings are genuinely recipe-invariant rather than interpolating recipe identifiers, we conduct two complementary generalization evaluations. First, we perform *leave-recipe-out* validation: we train $\hat{f}$ on runs from $\mathcal{R}_{\text{train}} \subset \mathcal{R}_{\text{exp}}$ and evaluate on a disjoint set of recipes $\mathcal{R}_{\text{test}}$, reporting prediction error for each $t \in \mathcal{T}$ and each held-out recipe. Second, we test *within-recipe* generalization by holding out runs (seeds) but not recipes. The gap between these two regimes quantifies the extent to which recipe drift dominates ordinary estimation error. We report per-task metrics (MSE, MAE, rank correlation over checkpoints) and also decision-centric metrics (below). When intervals $[\ell_t(Z), u_t(Z)]$ are produced, we report empirical coverage and interval width, both aggregated across tasks and stratified by recipe, to diagnose whether miscoverage is concentrated in particular recipe families.

OOD flagging evaluation. We operationalize the out-of-family decision $\text{OOD}(Z, r_{\text{new}})$ by constructing controlled "shift" recipes that violate H1 in interpretable ways, e.g., curricula that intentionally suppress a capability probed by some Q_j, or data mixtures that induce distributional changes in proxy coordinates not represented in $\mathcal{R}_{\text{train}}$. We then measure (i) false-positive rate on in-family held-out recipes, and (ii) detection power on these shifted recipes. Because OOD can be defined via both proxy-shift tests and residual patterns, we report both components and their conjunction/disjunction. A practical target is to tune thresholds so that in-family false positives are rare under run-wise exchangeability, while shifted recipes are flagged early in training (small C).

Early stopping and evaluation allocation as decision problems. We evaluate the usefulness of proxy-based prediction by treating checkpoint selection as a decision problem. Fix a target task t. Given a run i with proxy

trajectory $\{Z_{i,s}\}_s$, define the predicted best checkpoint

$$\hat{s}_t \;=\; \arg\max_{s\in\mathcal{S}} \hat{f}_t(Z_{i,s}),$$

and compare it to the oracle $s_t^\star = \arg\max_s Y_{i,s,t}$ among evaluated checkpoints. We report the *regret*

$$\mathrm{Regret}_t \;=\; Y_{i,s_t^\star,t} - Y_{i,\hat{s}_t,t},$$

averaged across runs and stratified by recipe. In addition, we evaluate an *interval-aware* policy that selects the earliest checkpoint s such that the predicted lower confidence bound exceeds a target threshold, thereby converting calibrated intervals into a compute-saving rule. This yields a direct estimate of compute saved (in terms of C or number of checkpoints evaluated) at fixed expected task performance.

Baselines: FLP and FLP-M. We compare against two intentionally constrained baselines that reflect common practice. FLP ("scalar loss predictor") uses only a scalar $S_{i,s}$ such as average validation loss/perplexity (or a small set of global losses aggregated into one number) and fits $g_t(S)$ to predict Y_t. FLP-M ("multi-loss predictor") uses the vector of domain losses $\{L_d\}_{d\in\Pi_{\mathrm{dom}}}$ but excludes targeted probes $\{Q_j\}$; it therefore tests whether probes add value beyond domain coverage. All predictors are trained with the same run-wise splitting and are calibrated with the same block-conformal procedure (i.e., the difference is the feature set, not the interval method). We additionally report a trivial baseline that predicts Y_t from training compute C alone, which quantifies how much of the signal is explained by time.

Ablations and stress tests. To isolate which parts of the proxy suite matter, we ablate (i) probes only, (ii) domains only, and (iii) individual proxy coordinates grouped by capability/domain. We also vary label sparsity (number of downstream evaluations per run) and the block length τ used in calibration, to confirm that interval validity is robust to reasonable choices of dependence modeling. Finally, we conduct a "bounded drift" sweep by continuously interpolating between two recipes (e.g., linearly mixing curricula schedules) and plotting prediction error as a function of recipe distance; this empirically probes whether degradation is graceful, as suggested by the bias term induced by drift.

Limitations and extensions. We emphasize that Hypothesis H1 is a modeling assumption rather than an identity, and the utility of PROXY-MANIFOLD depends on how well the proxy suite Π controls the relevant axes of variation induced by changing recipes. In particular, even when Π

is held fixed, changes to data mixture, curriculum ordering, tokenizer, context window, or optimization may alter $\mathbb{E}[Y_t \mid Z = z, R = r]$ in ways not representable as bounded drift. Thus, our guarantees should be read conditionally: when run-level exchangeability is plausible and when the mapping from proxies to tasks is approximately stable across the recipe family, we can obtain accurate predictions with calibrated uncertainty; when these conditions fail, we expect OOD flags or miscoverage, and the correct response is to enlarge Π, restrict $\mathcal{R}$, or increase downstream labeling.

Proxy sufficiency and unobserved confounding. A core limitation is that Z may be insufficient for Y across recipes: there may exist latent factors U affected by recipe choices that influence downstream behavior but are not captured by $\{L_d\}$ and $\{Q_j\}$. Formally, even if for each fixed recipe r we have $\mathbb{E}[Y_t \mid Z = z, R = r] = f_{t,r}(z)$, the existence of a recipe-invariant $f_t^\star$ requires $f_{t,r} \equiv f_t^\star$ for all r in-family. If $f_{t,r}$ varies in directions that do not induce detectable proxy shifts (cf. Thm 4), then residual-based OOD tests may have low power. This is the principal reason we advocate (i) a multi-coordinate suite Π that spans distinct capability axes, and (ii) explicit reporting of miscoverage stratified by recipe family, since average coverage can hide systematic failures concentrated on particular recipes.

Extensions to multimodal training. In multimodal settings (e.g., text–image, text–audio, or video), the natural extension is to treat Z as a concatenation of modality-specific and cross-modal proxies. Concretely, we may define

$$Z \;=\; \big[Z^{\text{text}},\; Z^{\text{img}},\; Z^{\text{aud}},\; Z^{\text{xmod}}\big],$$

where Z^{img} includes curated image-domain losses (captioning, VQA-style losses on fixed validation sets) and probe NLLs for visual grounding markers, while Z^{xmod} includes alignment probes (e.g., likelihood of correctly binding entities across modalities under controlled synthetic tasks). Two technical complications arise. First, proxy computation is no longer uniformly cheap: some multimodal probes require expensive encoders/decoders or large input resolutions, so the $O(K)$ cost model becomes heterogeneous and can dominate logging. Second, the definition of a "domain loss" L_d can be ambiguous across modalities (e.g., likelihood vs. contrastive losses), and we must ensure that the proxy coordinates remain comparable across recipes that change the multimodal objective. A practical extension is therefore to standardize Π in terms of fixed evaluation protocols (frozen preprocessing, fixed decoding, fixed scoring) rather than in terms of the training objective, so that Z remains an external measurement rather than an internal training artifact.

Extensions to tool-use and agentic traces. When models interact with tools (retrieval, code execution, browsers), a checkpoint induces not only to-

ken likelihoods but also distributions over action traces, which we may denote by A. A naive approach is to include in Π probe NLLs for tool-call syntax; however, syntactic compliance is not sufficient for functional tool use. We therefore consider trajectory-level proxies computed from a controlled environment with fixed prompts and deterministic tool backends: success rates, step counts, tool error incidence, and likelihood of reference actions under a teacher policy. One can represent these as additional coordinates Q_j^{tool} (e.g., negative log-likelihood of a canonical tool-call sequence, or a calibrated surrogate loss for environment reward). The main limitation is dependence: A is generated by closed-loop interaction, so checkpoint evaluations can have high variance and strong within-run correlation, requiring larger block lengths τ (or run-level aggregation) for valid intervals. Moreover, tool traces introduce a new kind of recipe shift: changes in system prompts, tool schemas, or decoding constraints can change $\mathbb{E}[Y \mid Z]$ without changing Z computed on static text domains, so Π must include probes that are invariant to these interface changes or explicitly include interface descriptors as covariates.

Interactions with data mixtures and curricula. Domain losses $\{L_d\}$ are themselves defined on curated datasets, which are typically proxies for (and not identical to) the pretraining mixture. If recipes vary data mixtures, then there are two distinct effects: (i) the marginal distribution of Z changes (covariate shift), and (ii) the conditional mapping $z \mapsto \mathbb{E}[Y_t \mid Z = z]$ may change (concept shift) because the same measured losses can correspond to different internal representations depending on what was emphasized during training. We view the bounded-drift model (Thm 2) as a first-order approximation of such effects, but it can be violated when curricula induce discrete phase changes (e.g., delayed emergence of in-context learning) or when mixture changes create capability trade-offs not tracked by Π. A principled extension is to augment Z with mixture descriptors $m(r)$ (e.g., the mixture weights over a fixed taxonomy, or summary statistics of the data stream) and to learn $f_t(z, m)$ with a regularizer that penalizes sensitivity to m within a trusted range. This does not eliminate drift, but it can convert an unmodeled shift into a modeled covariate.

Failure modes of prediction and calibration. We highlight several concrete failure modes. (i) *Proxy gaming:* if proxy datasets or probes are leaked into training, or if a recipe overfits them, then Z can improve without corresponding improvement in true downstream behavior, breaking H1 in a way that may evade proxy-shift tests. (ii) *Non-monotonic trajectories:* for some tasks, downstream performance can temporarily degrade even as most losses improve; monotonicity constraints, if imposed, can then induce biased predictions and overconfident intervals. (iii) *Task-dependent heteroskedasticity:* the residual variance σ^2 can depend on z (e.g., higher variance early in

training), in which case symmetric conformal intervals may be unnecessarily wide late in training or too narrow early. A straightforward remedy is to calibrate nonconformity scores that depend on Z (e.g., normalized residuals using a learned scale model), while preserving run-wise exchangeability in the calibration split.

Limitations of OOD flagging. Our OOD procedure is necessarily partial: a proxy-distribution test detects shifts in Z, and a residual test detects violations of the learned mapping $\hat{f}$ on labeled checkpoints, but neither detects all harmful shifts. In particular, if a new recipe changes downstream behavior while keeping both the proxy distribution and the residuals on the limited labeled set approximately unchanged, then any method based on $\mathcal{D}$ and Π will fail to flag (an identifiability limitation analogous to Thm 4). Accordingly, we treat $\mathrm{OOD}(Z, r_{\mathrm{new}})$ as a *screening* tool rather than a safety certificate. A practical extension is to include an adaptive labeling rule: when the proxy-shift p-value is small or when predicted intervals widen abruptly, we trigger additional downstream evaluations to increase the power of residual-based detection.

Governance implications and safe use. Because proxy-based prediction can reduce benchmark evaluation, it creates an incentive to substitute cheap proxies for expensive measurements. We regard such substitution as inappropriate for deployment decisions: $\hat{f}$ and its intervals are informative for *allocation* (what to evaluate next, where to early-stop, which recipes to prioritize), but they do not replace direct measurement of Y on the target tasks. Moreover, widespread sharing of Π and probe datasets can induce Goodhart effects, so any release should include guidance that probes are not training targets and should be monitored for contamination. Finally, reporting should explicitly separate (a) predictive performance within the in-family recipe set, (b) coverage diagnostics, and (c) OOD rates under specified shifts; without such stratification, there is a risk that aggregate numbers obscure systematic blind spots. In settings with high-stakes downstream impact, we recommend coupling PROXY-MANIFOLD with mandatory periodic full evaluations and with documentation of recipe changes, so that changes to the operational environment are treated as potential OOD events rather than silently absorbed by the predictor.

Artifact release: proxy harness, reference probes, evaluation scripts, and reporting standard. To make PROXY-MANIFOLD operational rather than purely conceptual, we release a complete artifact bundle whose purpose is to render the proxy vector Z and the associated inference pipeline ($\hat{f}$, $[\ell_t, u_t]$, OOD) reproducible across organizations and across time. Concretely, the bundle comprises: (i) a *proxy harness* that computes $Z_{i,s}$ from a

checkpoint, (ii) *reference datasets* defining the proxy suite Π (both curated domains $\{L_d\}$ and probes $\{Q_j\}$), (iii) *evaluation and calibration scripts* implementing training of $\hat{f}$, block-conformal interval construction, and OOD tests, and (iv) a *reporting standard* specifying what must be disclosed for proxy-based claims to be interpretable and comparable.

Proxy harness. The proxy harness is a deterministic evaluation layer which takes as input a checkpoint $\theta_{i,s}$, a proxy suite definition Π, and a configuration specifying decoding/scoring conventions, and returns a serialized proxy vector $Z_{i,s} \in \mathbb{R}^K$. Our design goal is that $Z_{i,s}$ be an *external measurement* rather than a byproduct of training, hence the harness fixes: tokenization, prompt templates, truncation rules, batching, precision, and reduction (mean NLL/perplexity). For each curated domain $d \in \Pi_{\text{dom}}$, the harness computes a validation loss $L_d(\theta_{i,s})$ on a pinned dataset snapshot with pinned preprocessing. For each probe $j \in \Pi_{\text{probe}}$, the harness computes $Q_j(\theta_{i,s})$, typically as an NLL on a synthetic or semi-synthetic set with a fixed format (e.g., tool-call syntax, retrieval markers, or structured outputs). We include a reference implementation for both left-to-right likelihood models and instruction-tuned chat models; in the latter case we standardize a "single-turn scoring" convention so that Q_j is well-defined even when the training objective is not pure next-token prediction. The harness emits, in addition to $Z_{i,s}$, an *evaluation manifest* containing software versions, dataset hashes, and an evaluation seed to support exact re-runs.

Reference datasets and probe definitions. We release the proxy suite Π as a versioned object consisting of (a) a list of domains d with dataset pointers and evaluation splits, and (b) a list of probes j with generation code and scoring code. For the domains, we favor small, contamination-resistant validation sets curated to span distinct axes (e.g., code, formal math, conversational instruction following, multilingual text), and we publish exact sample IDs and cryptographic hashes to prevent silent drift. For the probes, we release both the data (when static) and the generator (when procedurally defined), together with a formal specification of what constitutes a correct/target continuation. Since probes are especially vulnerable to Goodhart effects, we separate each probe into a *public* component (template family, scoring rule, intended capability axis) and an optional *withheld* component (private instantiations used only for auditing), and we document which results rely on which component. We further include lightweight contamination checks: given a training corpus snapshot, one can compute exact-match rates or approximate nearest-neighbor overlap against Π, and the harness will warn if overlap exceeds a declared threshold.

Evaluation scripts: fitting, calibration, and OOD. We provide scripts that reproduce the full pipeline from logs $\mathcal{D}$ to deployable predictors. Given a dataset $\mathcal{D} = \{(r_i, i, s, Z_{i,s}, Y_{i,s,\cdot})\}$, the scripts enforce run-wise splits (train/calibration/test) at the level of the index i to avoid leakage across correlated checkpoints. They implement the fitting step for several baseline model classes $\mathcal{F}$ (ridge, monotone GAMs, and shallow MLPs) and the invariance penalty described in the algorithmic template. For uncertainty, the scripts implement block-conformal calibration with user-specified block length τ, supporting both checkpoint-level blocks and run-level aggregation; the output is $[\ell_t(Z), u_t(Z)]$ for each $t \in \mathcal{T}$, together with empirical coverage diagnostics on held-out runs. For OOD, we include two complementary tests: (i) a proxy-shift test based on conformal p-values in Z-space (with options for learned embeddings or whitening), and (ii) a residual-based test that triggers when the new run exhibits an atypical frequency of interval violations on a small labeled subset. All scripts write a single machine-readable "model card" artifact containing Π version, $\hat{f}$ parameters, calibration quantiles q_t, and the chosen OOD thresholds, so that downstream consumers can apply the predictor without re-fitting.

Reproducibility, versioning, and governance of changes. Because the intended use is longitudinal (new recipes, new checkpoints), we treat Π and the harness as *versioned standards*. Every release includes semantic versioning, immutable dataset snapshots, and a compatibility table stating which harness versions can be compared without adjustment. When Π changes (e.g., a domain is replaced due to contamination), we require a deprecation period in which both $\Pi^{(v)}$ and $\Pi^{(v+1)}$ are computed for a shared window of runs, enabling an explicit mapping study of how predictions and intervals shift. We also provide a "minimal compliance" mode, intended for organizations unable to disclose full data, which permits reporting only Z, interval diagnostics, and hashed identifiers, while still allowing external auditors to verify that the proxy computation matches the standard.

Recommended reporting standard. To prevent proxy-based results from being selectively presented, we specify a reporting template that accompanies any claim using PROXY-MANIFOLD. At minimum, we require:

- **Proxy suite declaration:** the exact Π version, including domain/probe lists and any withheld components.
- **Recipe-family scope:** an explicit description of the in-family set $\mathcal{R}$ used to train/calibrate $\hat{f}$, including which axes (optimizer, context length, curriculum, tokenizer) were varied.
- **Run accounting:** the number of independent runs n, the checkpoint sampling policy, and the effective sample size estimate N_{eff} (including

the chosen τ).

- **Predictor specification:** the model class $\mathcal{F}$, training objective (including any invariance or monotonicity constraints), and hyperparameter selection protocol.
- **Interval diagnostics:** empirical coverage and average interval width for each t, reported both in aggregate and stratified by recipe (or recipe clusters), together with miscoverage concentration statistics.
- **OOD behavior:** proxy-shift p-values and residual-test outcomes on held-out recipe families or on declared shift scenarios, with false-positive/false-negative estimates at the chosen thresholds.
- **Downstream anchoring:** the subset of checkpoints at which $Y_{i,s,t}$ was directly measured for training/calibration, and the policy used to select these checkpoints.

We view these items as the minimal information needed for a reader to assess whether Hypothesis H1 is plausible for the stated scope and whether the reported uncertainty is meaningful. Finally, we recommend that all headline conclusions be accompanied by a "direct-eval" table on a small fixed set of checkpoints, to ensure that proxy-based extrapolations remain tethered to actual downstream measurements.