CONSERO: KNOWLEDGE-INFORMED PREDICTION FRAMEWORK WITH LANGUAGE AND TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models learn reusable predictive patterns, while domain knowledge describes which variables matter and how they interact. The challenge is to use such knowledge when it is incomplete or incorrect, grounding its predictive role in labeled observations. We propose Consero, a framework that connects language-proposed knowledge to a frozen tabular predictor by using declarations to guide evidence reading. A structured template distinguishes relevant, irrelevant, and unspecified features and records pairwise relationships; a knowledge–data encoder reads the corresponding support evidence to form a predictive correction. To learn this mapping, we pretrain the encoder on synthetic tasks that vary predictive mechanisms and knowledge quality, with prediction and semantic reconstruction objectives. For a new task, a language model revises declarations using prediction feedback from training examples, while the encoder and backbone remain fixed. Controlled synthetic experiments demonstrate predictive use of partial and fallible declarations and show how feature assignment affects the resulting gains. Across 156 real-world tasks, Consero improves mean classification macro-F1 and reduces normalized root mean squared error relative to frozen LimiX.

1 INTRODUCTION

Tabular prediction supports decisions in scientific, industrial, and business settings, where numerical measurements coexist with knowledge about what variables mean and how they relate. Tabular foundation models make numerical prediction reusable across tasks: prior-data-fitted networks (Müller et al., 2022) learn predictive priors from generated tasks, underpinning TabPFN (Hollmann et al., 2023a; 2025), TabICL (Qu et al., 2025), and TabICLv2 (Qu et al., 2026); LimiX (Zhang et al., 2025) extends conditional prediction through a large data model (LDM). Large language models (LLMs) can supply complementary domain hypotheses from task descriptions and feature meanings, as in CAAFE (Hollmann et al., 2023b). Connecting these hypotheses to a reusable predictor would let practitioners revise task knowledge while retaining the numerical model.

These hypotheses can enter prediction at different stages, as Figure 1 illustrates. On the left, priors shape synthetic pretraining data in (a), as in Do-PFN (Robertson et al., 2025) and CDFM (Qiao et al., 2026); in (b), models receive knowledge alongside observations, as in KODE (Xu et al., 2026) and KnowsTFM (Koloski et al., 2026). On the right, Consero (c) compiles language proposals into a template, uses it to read residual and pair evidence, and corrects a frozen predictor. The question is how an explicit interface can turn uncertain task knowledge into useful guidance from labeled examples.

Useful domain knowledge rarely specifies a complete prediction rule. In a manufacturing task, for example, a proposal might identify two process variables as jointly relevant, mark a record identifier as irrelevant, and leave other variables unspecified. These declarations tell a predictor what to examine, not what numerical answer to produce. Irrelevance must remain distinct from uncertainty, and a proposed relationship must be assessed using the corresponding observations. This motivates a division of labor: language supplies structured hypotheses, while labeled support examples determine how the numerical model can use them.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

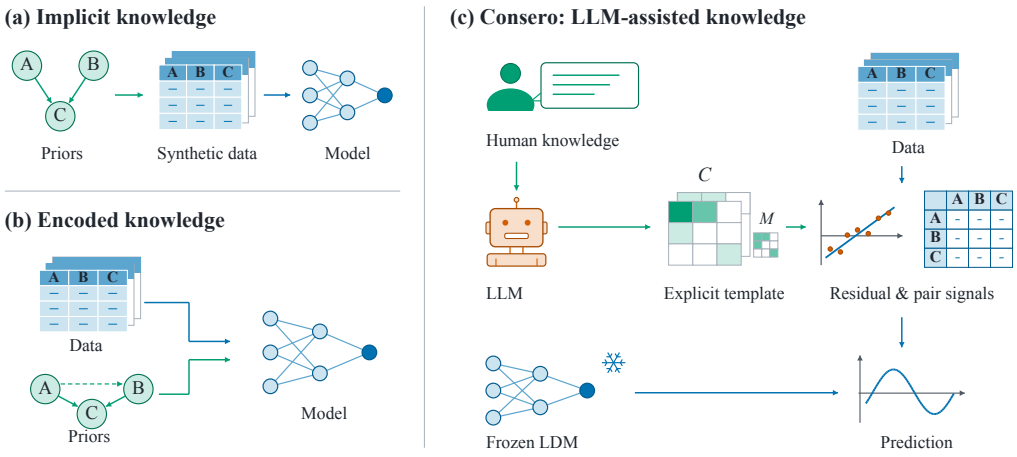


Figure 1: From domain knowledge to tabular prediction. (a) Knowledge shapes the pretraining distribution. (b) Explicit declarations condition a model. (c) Consero uses LLM-proposed knowledge to guide evidence reading around a frozen LDM. Green denotes knowledge, including the compiled template (C, M); blue denotes data and pair evidence.

We propose Consero, a knowledge-informed prediction framework that implements this division of labor (Figure 2). An explicit template preserves feature relevance, uncertainty, and proposed relationships. A pretrained knowledge–data encoder reads the template together with support evidence to form an external correction around a frozen LDM. Synthetic tasks teach the reader to use incomplete and conflicting declarations; downstream feedback revises the knowledge without retraining the reader or backbone. Our contributions are:

- We establish an executable interface for feature relevance, uncertainty, and relationships, providing a foundation for integrating language-based domain knowledge with numerical evidence in tabular prediction.
- We develop a pretrained knowledge–data encoder for prediction and knowledge revision around a frozen backbone. Controlled synthetic studies identify training scale and knowledge coverage as promising directions for further progress.
- On the full suite of 156 real tasks, Consero improves mean binary and multiclass macro-F1 by 2.32% and 3.59%, and reduces mean regression NRMSE by 1.85%, relative to frozen LimiX.

2 Related Work

Knowledge in pretraining distributions. Prior-data-fitted networks (Müller et al., 2022) amortize prediction under a task-generating prior; TabPFN (Hollmann et al., 2023a; 2025), TabICL (Qu et al., 2025), TabICLv2 (Qu et al., 2026), and LimiX (Zhang et al., 2025) develop this direction for tabular and conditional prediction. Do-PFN (Robertson et al., 2025) uses an SCM prior for interventional estimation, while CDFM (Qiao et al., 2026) uses mechanism assumptions and a variational formulation for discovery.

Semantic representations and transformations. A second line of work makes feature meaning available to the predictor. TabLLM (Hegselmann et al., 2023) serializes rows into text, while TP-BERTa (Yan et al., 2024) combines feature names with numerical tokens. CARTE (Kim et al., 2024) uses knowledge-base graph representations, TARTE (Kim et al., 2025) combines semantic row embeddings with residual boosting, and ConTextTab (Spinaci et al., 2025) incorporates semantics into in-context learning. Rather than changing the representation alone, CAAFE (Hollmann et al., 2023b) generates and evaluates feature-engineering code.

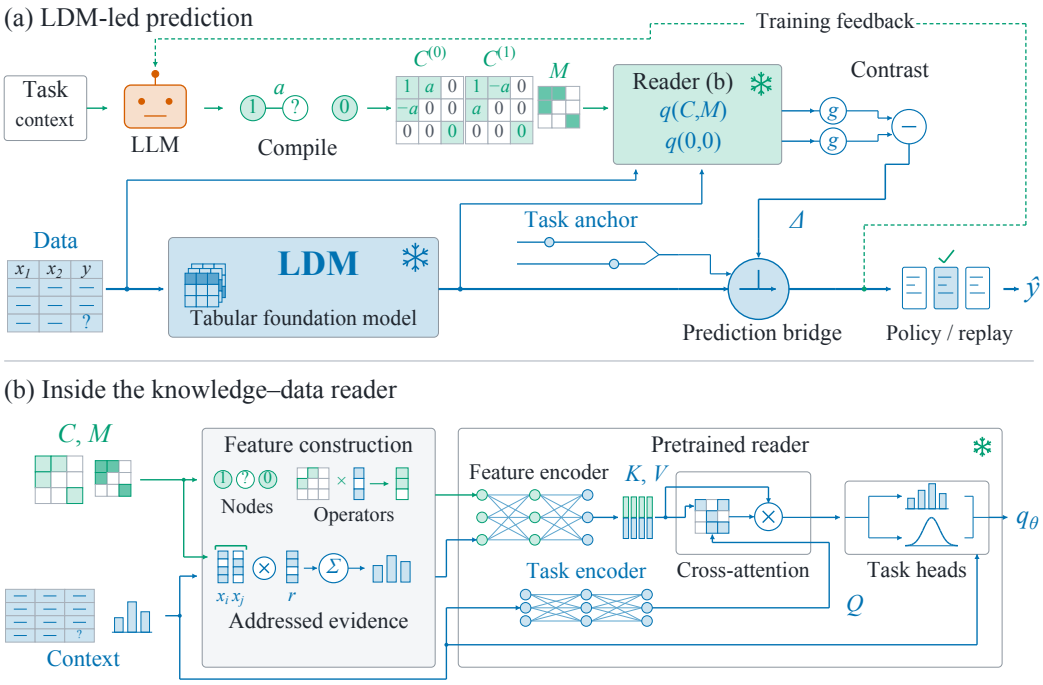


Figure 2: Consero framework. (a) LLM proposals guide matched readouts around a frozen LDM; training feedback revises the template. (b) Addressed evidence and node/pair declarations enter the pretrained reader. The bridge contrasts log probabilities or regression means (g). Green denotes knowledge, blue computation.

Explicit conditioning and adaptation. Explicit conditioning makes knowledge a model input, an integration strategy within informed learning (von Rueden et al., 2023). KODE (Xu et al., 2026) combines observations with weak knowledge for causal discovery; *Use What You Know* (Reuter et al., 2026) conditions causal effect estimation on partial ancestral graphs; KnowsTFM (Koloski et al., 2026) combines graph-derived attention priors with low-rank adaptation. Adapter tuning (Houlsby et al., 2019) and LoRA (Hu et al., 2022) adapt pretrained models through task-trained modules or low-rank weight updates. Consero uses an external encoder to read declarations against support evidence, with reader and backbone fixed downstream.

3 An Executable Semantic Interface

In Figure 2(a), the upper path runs left to right from task context through LLM proposals and compilation to readouts with and without knowledge. Their contrast joins the lower LDM path at the prediction bridge, followed by candidate selection and validation on training partitions. The dashed return path uses prediction feedback to revise knowledge while keeping model weights fixed. Panel (b) expands the reader below.

3.1 Representing Uncertain Domain Knowledge

The knowledge template gives language proposals a fixed, human-defined meaning. For support examples $S = \{(x_n, y_n)\}_{n=1}^{N}$ and query $x \in \mathbb{R}^F$, the reader $q_\theta(x \mid S, b, C, M)$ receives backbone predictions b, coordinates $C \in \mathbb{R}^{F\times F\times 2}$, and a known-state mask $M \in \{0,1\}^{F\times F}$. Addresses identify features, coordinates encode declarations, and the mask records whether a declaration is supplied. The reader comprises a shared feature encoder, task-conditioned attention, and prediction heads; the template specifies what these components read, while their weights are learned in pretraining.

Node declarations. The diagonal distinguishes three states:

$$(C_{ii,:}, M_{ii}) = \begin{cases} ((1,1),1), & \text{declared present,} \\ ((0,0),1), & \text{declared absent,} \\ ((0,0),0), & \text{unknown.} \end{cases} \tag{1}$$

Presence and absence concern proposed predictive relevance. The mask distinguishes explicit irrelevance from missing information even when both have zero coordinates. Node and pair declarations are independent, allowing a feature with unknown node status to participate in a proposed relationship.

Pair declarations. A sparse candidate specifies edges $\mathcal{E}$, signs $s_{ij} \in \{-1, 1\}$, and strength $\lambda \in \{0.35, 0.75, 1.15\}$. For each proposed pair, the compiler adds

$$a_{ij} = \frac{s_{ij}\lambda}{\sqrt{|\mathcal{E}|}}, \qquad C_{ij,:} \mathrel{+}= (a_{ij}, -a_{ij}), \quad C_{ji,:} \mathrel{+}= (-a_{ij}, a_{ij}), \quad M_{ij} = M_{ji} = 1. \tag{2}$$

The signed coordinates orient a pair operator; its predictive effect is learned from support evidence.

3.2 Reading declarations together with data

First-order summaries can hide joint behavior: Appendix A.1 constructs two tasks with identical marginal summaries but opposite interaction targets. We therefore combine template-dependent operator projections with residual–interaction statistics at the declared feature pairs. Let z denote features standardized using the support set. Residuals are $r_{nk} = \mathbf{1}[y_n = k] - b_k(x_n)$ for classification and scaled prediction errors for regression.

For an addressed pair $e = (i, j)$, let $\phi_{ne} = z_{ni} z_{nj}$ and define

$$u_{nek} = (\phi_{ne} - \bar{\phi}_e)(r_{nk} - \bar{r}_k), \qquad \mu_{ek} = \frac{1}{N}\sum_n u_{nek}, \qquad v_{ek} = \frac{\mathrm{sd}(u_{1ek}, \ldots, u_{Nek})}{\sqrt{N}}. \tag{3}$$

Bars denote support-set means. The statistic μ_{ek} measures how the addressed interaction covaries with the backbone's residual error, while v_{ek} summarizes its sampling variability. Multiplying μ_{ek} by the query's centered interaction $z_i z_j - \bar{\phi}_e$ carries this association to the current prediction. The reader averages these three quantities over edges incident to each feature. Thus, a declaration selects where to read, and observed residuals supply the response. Reversing a coefficient changes its operator projection while preserving the evidence at that address; moving an edge changes which interaction is examined.

Following Figure 2(b) from left to right, the feature encoder combines numerical evidence and pair operators with a separate node-state projection. The latter reads $d_i = (C_{ii,0} M_{ii}, M_{ii})$: the declared value and its known-state indicator, rather than the two coordinates of $C_{ii,:}$. Present, absent, and unknown thus become $(1, 1)$, $(0, 1)$, and $(0, 0)$. An irrelevance declaration changes the representation without removing the feature's observations.

The resulting tokens supply keys and values to cross-attention (Vaswani et al., 2017) on the right. The lower task-encoder branch forms queries from backbone class probabilities and support class frequencies for classification, or from backbone predictions and support residual summaries for regression. Task heads then read the attended states to produce class probabilities or a regression mean and scale. Reader weights are pretrained and frozen downstream; Appendix A specifies the architecture.

3.3 An external bridge around the LDM

The bridge converts the reader's response to knowledge into an adjustment of the backbone prediction. It compares two readouts with the same support examples and weights, changing only the template. For classification, this gives

$$\Delta_{\mathrm{cls}}(C, M) = \log q_\theta(C, M) - \log q_\theta(0, 0), \qquad p_C = \mathrm{softmax}\big(\log p_{\mathrm{raw}} + \Delta_{\mathrm{cls}}\big). \tag{4}$$

Arguments other than the template are suppressed. The contrast increases a class's weight when knowledge raises its reader probability; softmax normalizes the result. Regression instead uses the

mean difference $\Delta_{\text{reg}} = \mu_\theta(C, M) - \mu_\theta(0, 0)$ around a training-fitted affine anchor; classification likewise uses training-side decision adaptation. Empty knowledge gives zero contrast and recovers the matching anchor. Comparing against this anchor separates the knowledge contribution from task adaptation.

3.4 Training-side knowledge revision

Once a declaration can be evaluated through the bridge, prediction feedback can guide its revision. This follows the feedback-driven approach of Self-Refine (Madaan et al., 2023) and optimization by prompting (Yang et al., 2024), with numerical prediction providing the feedback here. Public task descriptions and feature metadata initialize a bank of eight LLM proposals. Each round compares candidate predictions with the adapted anchor, empty-template readouts, and corrupted-template controls on the same feedback examples. Aggregate loss differences and their uncertainty, together with classification error counts or regression bias, inform the next proposals. After three rounds, a policy partition selects a final-bank candidate or the unchanged anchor; a separate replay partition validates that choice before test scoring. Appendix D specifies the partitions and selection rules.

4 Model training

4.1 Synthetic data preparation

Training must connect declarations to their predictive use across tasks. As in KODE (Xu et al., 2026), we separate mechanism generation from knowledge sampling. Across three replicas, node warm-up and joint training cover 24,157 distinct episodes spanning three prediction tasks and four mechanism families (Table 1). Each target combines a base signal with an additional signal at a sampled feature pair; an analytic reference observes only the base signal, leaving the additional component for knowledge-conditioned correction.

Predictive mechanisms. Linear, multiplicative, piecewise, and saturating mechanisms vary the additional signal, which can also be locally active. Regression adds Gaussian noise; classification samples labels from signal-dependent probabilities, with $K \in \{3, 5, 10\}$ for multiclass tasks. Each episode has twelve features, 32, 128, or 512 support examples, and 128 queries. Training samples from 4,096 world indices per task, paired with zero and nonzero classification logit bias. Appendix B gives the generator and episode-counting rules.

Table 1: Distinct synthetic training episodes by task and mechanism across three replicas. An episode is indexed by world, support size, and classification bias within each task.

Task	Linear	Interaction	Piecewise	Saturation	Total
Binary	2,350	2,456	2,446	2,416	9,668
Multiclass	2,324	2,466	2,460	2,434	9,684
Regression	1,206	1,203	1,210	1,186	4,805
Total	5,880	6,125	6,116	6,036	24,157

Sampling fallible knowledge. Knowledge sampling varies what the reader is told about each generated task. Informative pairs address the additional signal, redundant pairs address base features, and conflicting pairs reverse orientation or select unrelated features. Node labels mark relevant features, including local gates. Positive-only and negative-only eligibility restrict node sampling to relevant and irrelevant features, respectively; mixed eligibility permits both. The three modes each have probability $1/3$; eligible declarations are retained with probability 0.75 and their values flipped with probability 0.2. Unobserved entries remain unknown, while clean labels are reserved for reconstruction supervision.

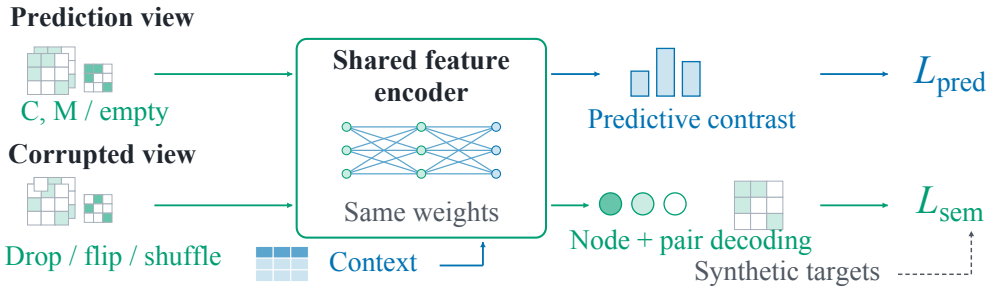


Figure 3: Shared-encoder pretraining. Separate prediction and corrupted-template forwards share feature-encoder weights. The corrupted view supports node and pair reconstruction; synthetic targets supervise decoding without entering the input.

4.2 Curriculum learning strategy

The curriculum introduces node semantics before joint reader training. For 2,048 updates, a zero-initialized node projection learns alongside fixed pretrained evidence and relation pathways. Another 2,048 updates train the full reader. Each update uses three tasks with two bias views each, giving 12,288 episode inputs per stage and replica. Knowledge conditions remain mixed while the trainable modules change. Three paired replicas share initialization and sampling schedules across objective variants; evaluation uses fixed final checkpoints.

4.3 Optimization objectives

Prediction alignment. As shown in the upper branch of Figure 3, knowledge-conditioned and empty-template views pass through the shared encoder to form the predictive contrast in Eq. 4. The analytic reference supplies the base prediction and the generator supplies the full conditional target. Classification supervises corrected probabilities using cross-entropy normalized by $\log K$ and $0.5(1 - \text{soft macro-F1})$. Regression supervises the contrast-adjusted mean using a scale-aware Gaussian risk.

Semantic reconstruction. The lower branch of Figure 3 applies a denoising objective (Vincent et al., 2010): dropped, flipped, or shuffled declarations pass through the same encoder before node and pair decoding. Both branches read support context; the lower branch uses synthetic truth as reconstruction targets, not as input. The two losses on the right combine as

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + 0.1\mathcal{L}_{\text{sem}}, \qquad \mathcal{L}_{\text{sem}} = \text{mean}\{\mathcal{L}_{\text{node}}, \mathcal{L}_{\text{edge}}\}, \tag{5}$$

with unavailable target groups omitted. Node and edge losses use squared error, with a bounded sign-preserving transform for edge targets. Appendix B specifies their normalization. Auxiliary heads are discarded downstream.

5 Experiments

5.1 Setup

The synthetic study evaluates knowledge interventions against an analytic reference and tests the frozen reader with official LimiX-16M predictions. Real-data experiments ensemble the same three pretrained readers around frozen LimiX-16M. Baselines include task-adapted LimiX, prediction-only training (*w/o reconstruction*), and initial proposals (*w/o revision*); Consero uses proposals after three revisions. The reconstruction control executes the same candidate trajectories. Appendices C and E detail the comparisons.

Metrics. We report macro-F1 and NRMSE, averaging synthetic episode scores or weighting real tasks equally. NRMSE divides RMSE by the sample label standard deviation, using support labels for synthetic tasks and full-training labels for real tasks.

5.2 SYNTHETIC DATA

Synthetic interventions separate how much knowledge is supplied from what its declarations say (Table 2). Correct supplies all node labels and the true pair; present-only retains relevant nodes and that pair; known-zero-only marks irrelevant nodes without edges. Empty removes all declarations, and fallible inputs follow training corruption. Observations and reader weights are fixed across these conditions. The analytic panel covers 24 held-out worlds per task across four training mechanism families, three support sizes, and three replicas (Appendix B).

Table 2: Synthetic prediction with analytic and official LimiX references. Panels use different evaluation worlds and support sizes; compare rows within each panel. Correct C uses synthetic ground truth. F1 is macro-averaged.

Reference / template	Binary F1 ↑	Multiclass F1 ↑	REG NRMSE ↓
Analytic reference: 24 worlds/task; support 32, 128, 512			
Analytic raw	0.64186	0.21704	0.72707
Unknown / empty	0.64186	0.21704	0.72707
Correct	0.68438	0.27348	0.37197
Fallible observed	0.67936	0.26684	0.37975
Present only	0.68523	0.27481	0.36956
Known zero only	0.66204	0.25252	0.46389
Correct, w/o reconstruction	0.68434	0.27353	0.37205
Fallible, w/o reconstruction	0.67930	0.26661	0.37975
Official LimiX-16M: 72 worlds/task; support 128			
LimiX-16M raw	0.65427	0.23555	0.46556
LimiX-16M + reader, correct C	0.68135	0.27276	0.40080
Relative change	**+4.14%**	**+15.80%**	**-13.91%**

Complete knowledge improves prediction in all three task types (Table 2). Binary macro-F1 rises from 0.64186 to 0.68438 and multiclass macro-F1 from 0.21704 to 0.27348, relative gains of 6.62% and 26.01%. Regression NRMSE falls from 0.72707 to 0.37197 (48.84%).

Partial declarations test whether prediction requires knowledge about every feature. Known-zero-only input improves both F1 means and reduces regression NRMSE from 0.72707 to 0.46389. Both this input and the empty template have zero coordinates, but their masks differ: one declares selected features irrelevant, while the other leaves them unspecified. The resulting prediction gap demonstrates the use of this distinction without pair declarations. Present-only input retains relevant nodes and the correct pair and slightly exceeds complete knowledge on all three means.

Fallible inputs retain relative F1 gains of 5.84% and 22.95% and an NRMSE reduction of 47.77% over analytic raw. Table 2 includes the matched prediction-only control; Appendix C reports the complete ablation.

We next vary declaration content to test which aspects of knowledge affect the correction. Figure 4 compares binary classification, multiclass classification, and regression from left to right. The top row measures the loss from removing knowledge; the middle and bottom rows test sign reversal and address shuffling. All nine mean contrasts favor correct templates. In each task, shuffling causes a larger mean loss than reversing signs: reassigning declarations changes which evidence is read, whereas sign reversal retains those locations.

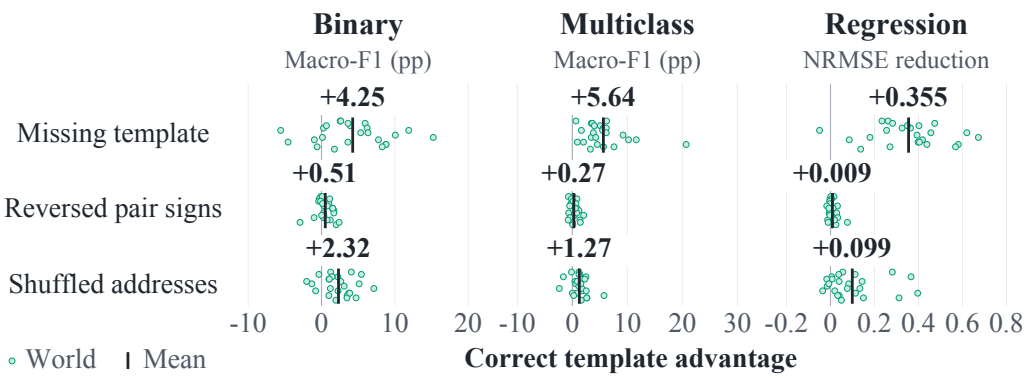


Figure 4: Sensitivity to synthetic knowledge content. Positive differences favor the correct template. Each point represents one of 24 paired worlds per task, averaged over three reader replicas and, for classification, both bias settings. Reader, support, and queries are fixed; address shuffling permutes node and pair declarations.

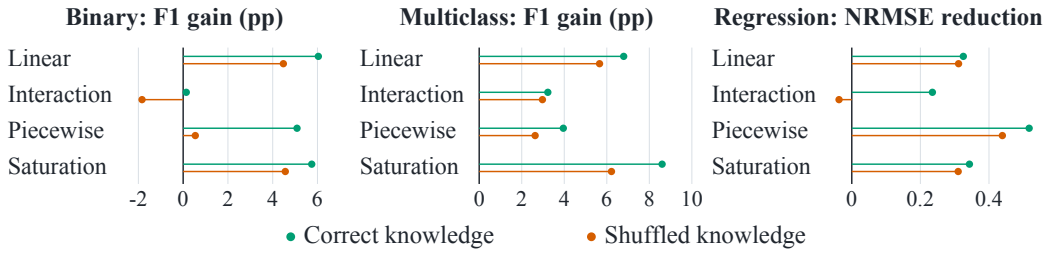


Figure 5: Knowledge effects by mechanism. Positive values improve on analytic raw. Each point averages six held-out worlds and three replicas, including both classification bias settings. Correct and shuffled templates use the same reader.

Figure 5 retains this task order and separates linear, multiplicative, piecewise, and saturating mechanisms from top to bottom. Green points lie to the right of zero in every panel: correct knowledge improves every task–mechanism mean. Comparing orange points, shuffled knowledge retains gains across all four multiclass mechanisms in the center, but crosses below raw for multiplicative regression on the right. There, NRMSE changes from 0.71197 to 0.47692 with correct knowledge and 0.74912 after shuffling. Feature assignment thus affects both the magnitude and direction of the correction's benefit.

To test transfer to a pretrained backbone, we evaluate official LimiX-16M on 72 synthetic worlds per task with 128 support and 128 query examples. Four-fold out-of-fold support predictions and full-support query predictions replace the analytic inputs to the frozen reader. With ground-truth declarations, binary and multiclass macro-F1 improve by 4.14% and 15.80%, and NRMSE falls by 13.91% (Table 2). Appendix C.1 gives the protocol and knowledge controls.

5.3 Real-world data

Datasets and protocol. We now replace generated mechanisms and supplied templates with real tasks and language proposals. We evaluate 71 binary and 35 multiclass tasks from BCCO-cls and 50 regression tasks from BCCO-reg, the benchmarks released with LimiX (Zhang et al., 2025), using fixed train/test splits and official LimiX-16M predictions. Public descriptions and feature names initialize GPT-6-Astra proposals, followed by the three-round protocol in Section 3.4. Classification uses task-adapted decision rules; regression uses a training-fitted affine anchor. Adapted LimiX applies the same rules without knowledge correction. Appendix D specifies the inference and evaluation protocol.

Table 3: Real-data classification. F1 is macro-F1; F1 and accuracy are percentages. Rel. % compares F1 with frozen LimiX. AUC is binary AUROC or multiclass macro OvO.

	Binary (71 tasks)				Multiclass (35 tasks)			
Model	F1↑	Rel. %	Acc.↑	AUC↑	F1↑	Rel. %	Acc.↑	AUC↑
Frozen LimiX	76.74	+0.00	84.19	0.8721	60.75	+0.00	68.23	0.8439
Consero	78.51	+2.32	83.20	0.8721	62.93	+3.59	67.46	0.8476

Table 4: Regression on 50 tasks. Δ and Rel. % are the absolute NRMSE change and relative reduction from frozen LimiX. Configurations match Table 3.

Model	NRMSE↓	Δ	Rel. %	R^2 ↑	Spearman↑
Frozen LimiX	0.4007	+0.0000	+0.00	0.7782	0.8595
Consero	0.3932	-0.0074	+1.85	0.7841	0.8597

Overall prediction. Relative to frozen LimiX, Consero raises binary and multiclass macro-F1 from 76.74/60.75 to 78.51/62.93 and reduces regression NRMSE from 0.4007 to 0.3932 (Tables 3 and 4). The relative improvements are 2.32%, 3.59%, and 1.85%, respectively; classification accuracy decreases by 0.99 and 0.77 percentage points.

Component comparisons. The full adaptation, reconstruction, and revision comparisons are reported in Appendix E; their task means remain close to adapted LimiX. Appendix F evaluates the node interface at warm-up on all 156 tasks; Appendix G compares training budget, knowledge sampling, and order. Appendix E reports complementary metrics and language-proposal comparisons.

Training scale and knowledge coverage. The synthetic studies also identify ways to strengthen the reader (Appendix G). Continuing training from 2,048 to 8,192 updates improves correct-knowledge prediction in all three task types and all three paired replicas. Both replaying existing tasks and adding fresh tasks yield these gains. At fixed budget, increasing exposure to sparse, correct declarations improves their use: binary and multiclass F1 rise by 1.62% and 4.53%, and NRMSE falls by 5.87%. These comparisons motivate studying training scale and knowledge coverage alongside backbone capacity. They also separate learning to interpret a declaration from obtaining useful declarations on a new task, connecting synthetic reader development to downstream knowledge acquisition.

From proposal to prediction. On BNG(echoMonths), an initial *still_alive–age* relationship is compiled into off-diagonal declarations and read against support evidence. Test NRMSE falls from 0.73743 for the adapted anchor to 0.72901 with this proposal (1.14%). The proposal selects a relationship to examine, while support observations determine its predictive use. This separation lets task knowledge change through an explicit declaration while preserving the learned reader and backbone. Appendix E.1 traces the selected knowledge, its encoding, and prediction.

6 Conclusion

This paper presents Consero, a framework that connects language-proposed knowledge to tabular prediction through an explicit semantic interface. Its pretrained reader interprets feature relevance and pairwise relationships using support evidence, producing corrections around a frozen backbone. Controlled synthetic experiments establish predictive use of complete, partial, and fallible declarations and show how feature assignment affects gains across mechanisms. Evaluation on 156 real-world tasks carries this interface to language-generated proposals, with prediction feedback revising knowledge rather than model weights.

AI USE STATEMENT

Generative AI tools were used solely to assist with language editing, clarity, and formatting during manuscript preparation. They were not used to conduct experiments, derive scientific results, or make research decisions. The authors reviewed all AI-assisted content and remain fully responsible for the scientific content, analysis, and conclusions.

REPRODUCIBILITY STATEMENT

The paper provides detailed descriptions of the model architecture, learning objective, datasets, baselines, evaluation metrics, and experimental settings to support reproducibility. The appendix further documents theoretical proofs, hyperparameter configurations, data-generation procedures, implementation details, and additional experimental results. Together, these materials are intended to provide sufficient information for reproducing the main results and evaluating the proposed method.

ETHICS STATEMENT

This work involves no human-subject experiments or collection of personally identifiable or sensitive information. All datasets are publicly available and used according to their terms. While the method introduces no risks beyond those generally associated with machine learning, its predictions may inherit data biases or errors. High-stakes deployment should therefore involve domain-specific validation and appropriate human oversight. We have considered research integrity, privacy, fairness, and potential misuse and are not aware of any violation of the ICLR Code of Ethics.

REFERENCES

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. TabLLM: Few-shot classification of tabular data with large language models. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 5549–5581. PMLR, 2023. URL https://proceedings.mlr.press/v206/hegselmann23a.html.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023a. URL https://arxiv.org/abs/2207.01848.

Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44753–44775. Curran Associates, Inc., 2023b. doi: 10.52202/075280-1938. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c2df4c35cdbee764ebb9e9d0acd5197-Abstract-Conference.html.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637:319–326, 2025. doi: 10.1038/s41586-024-08328-6. URL https://www.nature.com/articles/s41586-024-08328-6.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2790–2799. PMLR, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Myung Jun Kim, Léo Grinsztajn, and Gaël Varoquaux. CARTE: Pretraining and transfer for tabular learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 23843–23866. PMLR, 2024. URL https://proceedings.mlr.press/v235/kim24d.html.

Myung Jun Kim, Félix Lefebvre, Gaëtan Brison, Alexandre Perez-Lebel, and Gaël Varoquaux. Table foundation models: on knowledge pre-training for tabular learning. *arXiv preprint arXiv:2505.14415*, 2025. URL https://arxiv.org/abs/2505.14415.

Boshko Koloski, Xiangjian Jiang, Senja Pollak, Blaž Škrlj, Mateja Jamnik, and Nikola Simidjievski. KnowsTFM: Knowledge-informed fine-tuning of small tabular foundation models. *arXiv preprint arXiv:2606.30258*, 2026. URL https://arxiv.org/abs/2606.30258.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Jie Qiao, Ruichu Cai, Zijian Li, Weilin Chen, Pengfei Hua, Boyan Xu, Zhengming Chen, Zhifeng Hao, and Peng Cui. CDFM: Towards a general-purpose causal discovery foundation model. *arXiv preprint arXiv:2607.11508*, 2026. URL https://arxiv.org/abs/2607.11508.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50817–50847. PMLR, 2025. URL https://proceedings.mlr.press/v267/qu25d.html.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026. URL https://arxiv.org/abs/2602.11139.

Arik Reuter, Anish Dhir, Cristiana Diaconu, Jake Robertson, Ole Ossen, Frank Hutter, Adrian Weller, Mark van der Wilk, and Bernhard Schölkopf. Use what you know: Causal foundation models with partial graphs. *arXiv preprint arXiv:2602.14972*, 2026. URL https://arxiv.org/abs/2602.14972.

Jake Robertson, Arik Reuter, Siyuan Guo, Noah Hollmann, Frank Hutter, and Bernhard Schölkopf. Do-PFN: In-context learning for causal effect estimation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 174811–174848. Curran Associates, Inc., 2025. doi: 10.52202/085713-5814. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/ff997469ac66cf893c4183efeb22212a-Abstract-Conference.html.

Marco Spinaci, Marek Polewczyk, Maximilian Schambach, and Sam Thelin. ConTextTab: A semantics-aware tabular in-context learner. In *Advances in Neural Information Processing Systems*, volume 38, pp. 147005–147040. Curran Associates, Inc., 2025. doi: 10.52202/085713-4918. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/d807e7678ba3afd3a904f4af52819e77-Abstract-Conference.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(110):3371–3408, 2010. URL http://jmlr.org/papers/v11/vincent10a.html.

Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, Michal Walczak, Jochen Garcke, Christian Bauckhage, and Jannis Schuecker. Informed machine learning—a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):614–633, 2023. doi: 10.1109/TKDE.2021.3079836. URL https://doi.org/10.1109/TKDE.2021.3079836.

Wenbo Xu, Yue He, Yunhai Wang, Xingxuan Zhang, Kun Kuang, Yueguo Chen, and Peng Cui. A knowledge-informed pretrained model for causal discovery. *arXiv preprint arXiv:2603.20842*, 2026. URL https://arxiv.org/abs/2603.20842.

Jiahuan Yan, Bo Zheng, Hongxia Xu, Yiheng Zhu, Danny Z. Chen, Jimeng Sun, Jian Wu, and Jintai Chen. Making pre-trained language models great on tabular prediction. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=anzIzGZuLi.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, Ningbo Dai, Renzhe Xu, Shuyang Li, Tianyang Zhang, Yue He, Yuanrui Wang, Yunjia Zhang, Zijing Xu, Dongzhe Li, Fang Gao, Hao Zou, Jiandong Liu, Jiashuo Liu, Jiawei Xu, Kaijie Cheng, Kehan Li, Linjun Zhou, Qing Li, Shaohua Fan, Xiaoyu Lin, Xinyan Han, Xuanyue Li, Yan Lu, Yuan Xue, Yuanyuan Jiang, Zimu Wang, Zhenlei Wang, and Peng Cui. LimiX: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025. URL https://arxiv.org/abs/2509.03505.

A TEMPLATE AND ENCODER DETAILS

Figure 6 follows compilation from left to right. The three node states at the upper left fill the diagonals of $C^{(0)}$ and $C^{(1)}$; the pair below fills their off-diagonal coordinates. The rightmost mask distinguishes the zero-valued unknown node x_2 from the zero-valued absent node x_3. Its off-diagonal entries also retain the declared pair involving x_2, so an unknown node state does not erase a known relationship.

To read this template with the data, the feature encoder combines seventeen inputs. Seven numerical channels summarize standardized query values, marginal residual evidence, and task-dependent interactions. Five off-diagonal operator channels provide two outgoing projections, two incoming projections, and mask degree; node diagonals are excluded. Three residual–interaction channels supply association, query response, and uncertainty at declared addresses. These fifteen channels enter an affine projection to 32 dimensions. A separate bias-free projection maps the two node channels, value-times-known and known state, to the same dimension. The summed states pass through normalization, activation, and a linear map. Two four-head cross-attention layers use these feature states as keys and values and encoded seven-dimensional task inputs as queries.

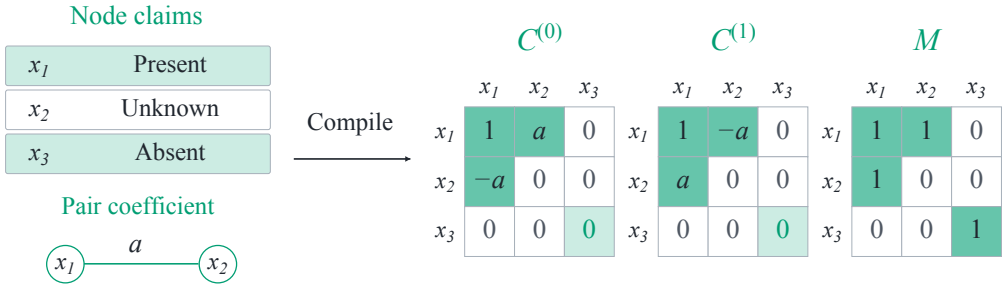


Figure 6: A schematic knowledge template. Diagonals declare x_1 present, x_2 unknown, and x_3 absent; the independent pair has coefficient a. Green cells are known, including zeros; white cells are unknown. Thus absence and uncertainty remain distinct.

Task inputs and prediction heads. The seven classification task inputs are the raw class probability, its logarithm, the support class frequency, predictive entropy, maximum raw probability, the gap between the largest two probabilities, and $\log K$. Regression uses the standardized raw prediction, its magnitude, support residual mean and standard deviation, and three constants $(1, 0, 0)$. The task encoder uses a linear projection, layer normalization, GELU, and a second linear map. Cross-attention blocks include residual connections, normalization and feed-forward updates. The shared classification head emits one correction per class token; these are centered and added to raw log probabilities before a joint softmax. Regression predicts a support-scale-adjusted mean correction and a softplus scale with a positive offset. These task-native heads use the same feature representation as the auxiliary decoders.

A.1 A FINITE-SUPPORT INTERACTION EXAMPLE

Consider all four combinations of $(x_1, x_2) \in \{-1, 1\}^2$, repeated equally, with zero base prediction and identical node declarations and addressed pair. Define $y^{(+)} = x_1 x_2$ and $y^{(-)} = -x_1 x_2$. Both tasks have mean zero, population variance one, and $\mathbb{E}[y x_1] = \mathbb{E}[y x_2] = 0$. A predictor whose label-dependent inputs are restricted to residual mean, residual scale and first-order residual–feature correlations therefore receives identical inputs at any fixed query. If its common output is a and the positive-task target is $t \in \{-1, 1\}$, its average squared error across the two tasks satisfies

$$\tfrac{1}{2}\left[(a - t)^2 + (a + t)^2\right] = a^2 + 1 \geq 1. \tag{6}$$

In contrast, $\mathbb{E}[y^{(+)} x_1 x_2] = 1$ and $\mathbb{E}[y^{(-)} x_1 x_2] = -1$, so a residual–interaction moment distinguishes them. The equal positive residual normalization preserves this distinction. On this balanced finite support, the addressed interaction resolves the ambiguity left by the specified first-order summaries.

B SYNTHETIC GENERATION AND TRAINING DETAILS

Episode and curriculum settings. Each episode contains twelve features, 128 queries, and 32, 128, or 512 support examples. Each update samples one world index and support size per task, generating paired bias views and averaging losses equally across tasks. Node warm-up and joint training each run for 2,048 updates from pretrained evidence and relation weights. Within each replica, both stages and the matched training objectives reuse the same schedule. They process $2{,}048 \times 3 \times 2 = 12{,}288$ episode inputs per stage; the two nominal regression bias views contain identical data.

Training data volume. Episodes are generated on demand from 4,096 candidate world indices per task, with 1,024 indices per mechanism. Across three replicas, node warm-up and joint training visit 3,209 binary, 3,199 multiclass, and 3,179 regression world indices. Table 1 reports the union of their task–world–support–bias combinations, counting the identical regression bias views once. This yields 24,157 distinct training episodes; the held-out panels are specified below and in Appendix C.1.

Covariates and conditional signals. Support and query covariates are sampled independently from standard normal distributions before setting $x_{12} = 0.65x_3 - 0.45x_4 + 0.5\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$. A true pair (i, j) is sampled from feature indices 3 through 11. Let $a \in \{-1, 1\} \times \{0.35, 0.75, 1.15\}$. The shared base signal is

$$g(x) = a\{0.8x_1 - 0.6x_2 + 0.25\tanh(x_1 + x_2)\}. \tag{7}$$

The additional signal is $h(x) = af(x_i, x_j)$, using one of four families:

$$f(u, v) = \begin{cases} 0.9u - 0.7v & \text{linear,} \\ 1.1uv & \text{interaction,} \\ 1.2\{u\mathbf{1}[u + v > 0] - v\mathbf{1}[u + v \le 0]\} & \text{piecewise,} \\ 1.4\tanh(u - 0.8v) & \text{saturation.} \end{cases} \tag{8}$$

In local worlds, h is multiplied by $\mathbf{1}[x_{12} > 0]$. Regression samples $y = g(x) + h(x) + \eta$ with $\eta \sim \mathcal{N}(0, 0.2^2)$, while the reference prediction is $g(x)$. The additional signal is therefore a known target for predictive correction.

For classification, let w be a random permutation of K evenly spaced values from -1 to 1, and let v be the centered cosine of K evenly spaced angles from 0 to π. The target and reference probabilities are

$$\begin{aligned} p^\star(y = k \mid x) &= \mathrm{softmax}_k\{g(x)w + h(x)v + \beta\}, \\ p_{\mathrm{raw}}(y = k \mid x) &= \mathrm{softmax}_k\{g(x)w + \beta\}. \end{aligned} \tag{9}$$

Labels are sampled from $p^\star$. The zero-bias setting uses $\beta = 0$; the nonzero-bias setting uses K evenly spaced values from 1.2 to -1.2. The generator does not impose exact class counts. Regression has no class-bias term, so the two nominal prior settings yield identical episodes.

Knowledge sampling and evaluation interventions. The true off-diagonal template stores the signed coordinates of the additional signal. The observed edge may instead point to the two base features, reverse sign, or use unrelated addresses. Node targets mark base features, the true pair, and any local gating feature as relevant. During training, declarations are drawn uniformly from positive-only, negative-only, and mixed eligibility; each eligible declaration is retained with probability 0.75 and its value flipped with probability 0.2. The auxiliary view uses a second corruption: known node masks are dropped with probability 0.25, node values flipped with probability 0.2, and one of four edge operations is sampled uniformly (retain, remove, sign-reverse, address-permute). All operations act on observed declarations; only the reconstruction targets consult synthetic truth.

At evaluation, nine template conditions isolate the availability and content of knowledge, with analytic raw as the reference. Correct input combines all true node labels with the true pair. Present-only removes negative node declarations but keeps the correct pair; known-zero-only removes all edges and all positive node declarations. Node flip, edge sign flip, and their combination retain the respective masks. Shuffle permutes both feature axes of the whole template and mask. Unknown clears both objects. Fallible observed uses the same node-corruption procedure and information regimes as training.

Optimization and reproducibility. The classification objective is soft-target cross-entropy divided by $\log K$, plus $0.5(1-\text{soft macro-F1})$. Regression uses Gaussian risk for the contrast-adjusted mean against the conditional mean, with a positive predicted scale and support-label normalization. Reconstruction pools feature states over query and task-token axes. A sigmoid node head predicts relevance; a tanh edge head reads concatenated ordered feature states. For signed coordinate targets, $\psi(c) = c/(1+|c|)$ preserves sign and bounds magnitude. With synthetic node labels $d_i^\star$ and declared off-diagonal target set $\mathcal{I}_\star$,

$$\mathcal{L}_{\text{node}} = \frac{1}{F}\sum_i (\hat{d}_i - d_i^\star)^2, \tag{10}$$

$$\mathcal{L}_{\text{edge}} = \frac{1}{4(2|\mathcal{I}_\star|)} \sum_{(i,j)\in\mathcal{I}_\star} \sum_{c=0}^{1} \big(\hat{e}_{ijc} - \psi(C^\star_{ijc})\big)^2. \tag{11}$$

Unlabeled edges are excluded rather than assigned zero, and empty target groups are omitted from the semantic-loss mean.

Three paired replicas continue from the same pretrained node-aware initialization within each pair. The optimizer is AdamW (Loshchilov & Hutter, 2019) with learning rate 3×10^{-4} and weight decay 10^{-4}; the gradient norm is clipped to 2. All reader prediction modules are trainable, with dropout disabled. Semantic decoding uses a 32-to-1 node head and a 64-to-2 ordered-pair head. Prediction-only training excludes both semantic heads from optimizer updates and evaluates the auxiliary view without gradients. Both arms use the final step-2,048 checkpoint.

Reconstruction gradients reach the prediction-shared feature encoder. Appendix C compares held-out predictions with and without this training objective.

The held-out panel reuses 24 worlds across replicas: six per mechanism and eight per support size, with five redundant, nine informative, five local, and five conflicting worlds per task. Multiclass class count and support size share the world-index cycle. Classification averages 48 world/bias instances per replica; regression has 24 distinct instances because its bias settings are identical.

C Additional synthetic results

C.1 Transfer to official LimiX predictions

This comparison changes the source of predictions supplied to the reader. It reuses 72 previously evaluated synthetic worlds per task, each with 128 support and 128 query examples, and replaces analytic predictions with official LimiX outputs. Two class-prior views for classification and one for regression yield 360 prediction inputs. This panel is separate from the 24-world analytic evaluation.

The official LimiX-16M checkpoint uses the unmodified predictor and default retrieval configurations. Label-independent four-fold partitions produce support predictions, excluding held-out labels from fitting and regression normalization; query predictions use the complete support set. Regression outputs are restored to their original scale, and NRMSE uses the support-label sample standard deviation. Both reference inputs are replaced before applying the three frozen reader replicas, with unchanged templates and execution strength. This panel uses the reader alone, without task adaptation, selection, or language revision.

Table 5 compares synthetic ground-truth templates with empty, wrong, and mixed-edge inputs. Wrong reverses node declarations and pair signs but retains true pair addresses; mixed edges can retain base-feature relations. Empty knowledge recovers raw predictions.

C.2 Matched reconstruction and analytic-reference controls

Matched reconstruction ablation. Table 6 compares prediction-only and reconstruction-trained readers under every template condition. Three paired replicas share initialization, world schedules, optimizer, corruption views, and the 2,048-step budget. The prediction-only control excludes auxiliary gradients and head updates; both arms use fixed final checkpoints. Template-versus-empty gains are present under both objectives, with smaller, task-dependent differences from reconstruction.

Table 5: Knowledge controls with official LimiX-16M predictions. All rows use the same synthetic panel and frozen reader replicas. Wrong templates retain true pair addresses.

Template	Binary F1 ↑	Multiclass F1 ↑	NRMSE ↓
Raw / unknown	0.65427	0.23555	0.46556
Correct	0.68135	0.27276	0.40080
Wrong	0.67085	0.26285	0.39616
Mixed edges	0.66715	0.25395	0.42084

Corrupted-template controls. Node flips, pair-sign reversal, and address shuffling test distinct parts of the template while keeping the reader fixed. Tables 7 and 8 organize these comparisons by mechanism and class bias. Present-only and known-zero-only compare different subsets of available declarations.

Complete results. Tables 6–9 report complete interventions, subgroup results, and complementary classification metrics using the main-text aggregation rules.

Table 6: Synthetic template interventions under prediction-only training and prediction plus reconstruction. Raw and empty inputs share identical primary scores. Shading identifies the correct-template reference.

	Binary F1 ↑		Multiclass F1 ↑		NRMSE ↓	
Template	Pred. only	+ Recon.	Pred. only	+ Recon.	Pred. only	+ Recon.
Raw / empty	0.64186	0.64186	0.21704	0.21704	0.72707	0.72707
Correct	0.68434	0.68438	0.27353	0.27348	0.37205	0.37197
Fallible observed	0.67930	0.67936	0.26661	0.26684	0.37975	0.37975
Present only	0.68507	0.68523	0.27422	0.27481	0.36961	0.36956
Known zero only	0.66215	0.66204	0.25234	0.25252	0.46394	0.46389
Node flip	0.67404	0.67398	0.26814	0.26821	0.38334	0.38344
Edge sign flip	0.67945	0.67927	0.27106	0.27078	0.38085	0.38076
Node + edge flip	0.67113	0.67167	0.26589	0.26651	0.39479	0.39489
Shuffled	0.66126	0.66122	0.26093	0.26074	0.47117	0.47121

Table 7: Mechanism-wise results for the reconstruction-trained reader. Each task panel compares the same four knowledge conditions across six held-out worlds per mechanism, reused across replicas. Fallible denotes the observed, corrupted template.

Mechanism	Raw	Correct	Fallible	Shuffled
Binary macro-F1 ↑				
Linear	0.64038	0.70077	0.69353	0.68514
Interaction	0.66019	0.66158	0.65901	0.64182
Piecewise	0.62272	0.67358	0.66723	0.62814
Saturation	0.64414	0.70158	0.69767	0.68977
Multiclass macro-F1 ↑				
Linear	0.20957	0.27751	0.26149	0.26617
Interaction	0.22262	0.25489	0.25313	0.25235
Piecewise	0.22210	0.26166	0.26113	0.24840
Saturation	0.21385	0.29986	0.29162	0.27605
Regression NRMSE ↓				
Linear	0.68092	0.35575	0.37961	0.36959
Interaction	0.71197	0.47692	0.47623	0.74912
Piecewise	0.84111	0.32398	0.33559	0.40209
Saturation	0.67429	0.33126	0.32757	0.36406

Table 8: Classification results under zero and nonzero generator logit bias. Paired columns compare the bias settings within each task; raw and empty inputs share identical F1 scores.

	Binary F1 ↑		Multiclass F1 ↑	
Template	Zero bias	Nonzero bias	Zero bias	Nonzero bias
Raw / empty	0.68694	0.59677	0.20589	0.22819
Correct	0.73323	0.63552	0.29726	0.24970
Fallible observed	0.73502	0.62370	0.28726	0.24643
Present only	0.73315	0.63731	0.29766	0.25196
Known zero only	0.71960	0.60448	0.26877	0.23627
Node flip	0.72660	0.62136	0.29142	0.24501
Edge sign flip	0.73429	0.62424	0.29373	0.24783
Node + edge flip	0.72711	0.61623	0.28930	0.24372
Shuffled	0.70761	0.61482	0.28082	0.24067

Table 9: Complementary classification metrics for the reconstruction-trained reader. Soft-target NLL is cross-entropy against the generator's conditional class probabilities.

Task	Template	Accuracy ↑	Bal. accuracy ↑	Soft-target NLL ↓
Binary	Analytic raw	0.76335	0.64712	0.50642
	Correct	0.79525	0.68440	0.43983
	Fallible observed	0.79546	0.67926	0.44367
	Shuffled	0.77924	0.66224	0.46579
Multiclass	Analytic raw	0.40592	0.28590	1.46361
	Correct	0.41390	0.30320	1.44271
	Fallible observed	0.41227	0.29928	1.44855
	Shuffled	0.40522	0.29209	1.46216

D Downstream inference and evaluation protocol

Data sources. The evaluated tasks use the BCCO-cls and BCCO-reg releases associated with LimiX (Zhang et al., 2025), retaining their archived train/test splits.[1]

Proposal and selection protocol. Public task descriptions, ordered feature names, and output vocabulary initialize eight candidates per semantic or anonymous bank. No sample rows or observed labels enter the task cards. Three revision rounds compare candidates with anchor, empty-template, and corrupted-template responses on training-side feedback partitions. Only the final bank and identity action enter policy/replay selection. The reconstruction-trained reader drives revision; the prediction-only control evaluates those same candidate trajectories. The initial-bank comparison keeps reader weights fixed and removes revision.

Selection and validation. The policy partition ranks these candidates against the adapted predictor. For classification, selection maximizes

$$J = \Delta\,\mathrm{MacroF1} + 0.5\Delta\,\mathrm{BalancedAccuracy} - 0.1\,\max(\Delta\,\mathrm{LogLoss}, 0),$$

where each difference is relative to adapted identity. The replay partition accepts a nonidentity action when $J > 0$. Regression instead selects by policy mean squared error and requires lower replay error than identity. Feedback, policy, and replay partitions are disjoint within integration; backbone contexts and task adaptation retain shared upstream dependencies. Test labels are used only after decisions are fixed.

Metrics and dependencies. Classification averages macro-F1 equally across tasks. Regression NRMSE uses full-training-label standard deviation with $\mathrm{ddof} = 1$; synthetic NRMSE uses support labels. The requested LLM identifier is `gpt-6-astra`. Evaluation records 775 successful

[1]Dataset releases: `https://huggingface.co/datasets/stableai-org/bcco_cls` and `https://huggingface.co/datasets/stableai-org/bcco_reg`.

responses: 310 initial responses including anonymous controls, and 155 per revision round. One single-feature task uses identity without language generation.

Practical implementation. Unknown and known-zero cells remain distinct throughout compilation. The two readout branches share support data and reader weights, making the empty-template correction zero. Classification ensembles average branch probabilities before taking their log contrast. Auxiliary reconstruction heads are discarded for downstream inference.

E ADDITIONAL REAL-TASK RESULTS

These results complement the task means in Tables 3 and 4, using the inference protocol in Appendix D.

Adaptation and component comparisons. Tables 10 and 11 give the full comparisons for the same reader and task splits as the main tables. Adapted LimiX uses the same training-side decision adaptation or affine anchor without knowledge correction. The reconstruction control uses the same proposal trajectories; the revision control evaluates the initial semantic bank with the same trained reader. Mean scores remain close to adapted LimiX, and reconstruction and revision produce small task-average changes.

Table 10: Full classification comparisons on 71 binary and 35 multiclass tasks. F1 and accuracy are percentages; relative F1 changes use frozen LimiX as reference. AUC denotes binary AUROC or multiclass macro OvO.

	Binary (71 tasks)				Multiclass (35 tasks)			
Model	F1↑	Rel. %	Acc.↑	AUC↑	F1↑	Rel. %	Acc.↑	AUC↑
Frozen LimiX	76.74	+0.00	84.19	0.8721	60.75	+0.00	68.23	0.8439
Adapted LimiX	78.51	+2.32	83.20	0.8721	63.00	+3.71	67.47	0.8480
w/o reconstruction	78.51	+2.31	83.20	0.8721	62.93	+3.59	67.46	0.8476
w/o revision	78.51	+2.32	83.20	0.8721	62.93	+3.59	67.46	0.8476
Consero	78.51	+2.32	83.20	0.8721	62.93	+3.59	67.46	0.8476

Table 11: Full regression comparisons on 50 tasks. Absolute and relative NRMSE changes use frozen LimiX as reference. Configurations match Table 10.

Model	NRMSE↓	Δ	Rel. %	R^2 ↑	Spearman↑
Frozen LimiX	0.4007	+0.0000	+0.00	0.7782	0.8595
Adapted LimiX	0.3934	-0.0072	+1.80	0.7838	0.8595
w/o reconstruction	0.3932	-0.0074	+1.85	0.7841	0.8597
w/o revision	0.3932	-0.0074	+1.86	0.7841	0.8597
Consero	0.3932	-0.0074	+1.85	0.7841	0.8597

Complementary metrics. The multiclass comparison in Table 3 also involves probability quality. Relative to frozen LimiX, log loss changes from 0.7642 to 0.7913 and Brier score from 0.4030 to 0.4152; lower is better for both metrics.

Language metadata. To examine the information supplied to the LLM, we compare initial proposals generated from anonymous and semantic metadata. Their respective macro-F1 scores are 0.785178 and 0.785126 for binary classification, and 0.630259 and 0.629259 for multiclass classification. Regression NRMSE is 0.393540 and 0.393226, respectively.

Declaration coverage. Initial proposals contain 61 known-present entries and no known-absent entries. Later proposals also contain no known-absent nodes, so full and positive-only controls coincide for these generated banks.

E.1 A SINGLE-ROUND LLM-FIRST COLLABORATION

Selected knowledge. The BNG(echoMonths) example in Section 5.3 follows one initial proposal through compilation and prediction. The LLM supplies eight initial proposals, from which training-side selection retains a tentative relationship between *still_alive* and *age*: their status and age information may jointly help predict the continuous target. The task card supplies feature names but leaves the target meaning unspecified. All nine node declarations remain unknown.

Template encoding. Using one-based feature indices, x_1 is *still_alive* and x_2 is *age*. The selected proposal supplies one pair with sign $+1$ and strength 0.35. Equation 2 gives

$$\begin{aligned} C_{12,:} &= (0.35, -0.35), \quad C_{21,:} = (-0.35, 0.35), \\ M_{12} &= 1, \qquad\qquad\quad M_{21} = 1. \end{aligned} \tag{12}$$

All other coordinates and mask entries are zero. Thus, the knowledge occupies the off-diagonal pair category; $C_{ii,:} = (0, 0)$ with $M_{ii} = 0$ encodes unknown node relevance, not declared absence. The signs orient the pair operator rather than prescribe a causal or target-effect direction.

Evidence-conditioned prediction. The frozen reader contrasts this template with empty knowledge on the same support evidence, correcting the fitted affine anchor. The candidate passes training-side policy selection and replay validation before test scoring. On 5,249 test examples, NRMSE is 0.76206 for frozen LimiX, 0.73743 for adapted LimiX, and 0.72901 with the initial proposal. The additional reduction over adapted LimiX is 0.00842 (1.14%). This selected case demonstrates an effective LLM-first collaboration: a language-proposed relation, made executable through C, improves prediction when interpreted against support evidence by the pretrained reader.

F NODE-INTERFACE ABLATION

We isolate the node interface at the warm-up stage of Section 4.2, before joint training. The pre-trained pair-evidence pathway and prediction heads remain fixed while the 64 parameters of the additive node projection are trained for 2,048 updates in each of three replicas. Removing node declarations then leaves the pair-only readout unchanged. The resulting readers are ensembled on the same 71 binary, 35 multiclass, and 50 regression tasks using frozen LimiX-16M, the task-adapted anchor, and training-side selection.

This comparison uses a fixed bank of initial GPT-5.5 relationship proposals. Features appearing as endpoints receive positive node declarations; other nodes remain unknown. The pair-only and node-enabled rows share these proposals. Anonymous and mismatched-metadata controls change the proposal source; the address control permutes only node declarations. Table 12 reports the resulting predictions after replay validation.

Table 12: Node-interface ablation at the warm-up checkpoint on all 156 real tasks. F1 is macro-F1 on the $[0, 1]$ scale. Node declarations are derived from proposal endpoints; the shaded row is the reference for the three semantic controls.

Configuration	Binary F1 ↑	Multiclass F1 ↑	NRMSE ↓
Frozen LimiX	0.767361	0.607451	0.400671
Adapted LimiX	0.785143	0.629966	0.393439
Consero, no node declarations	0.785010	0.630392	0.393695
Consero, endpoint-derived nodes	0.785513	0.629835	0.393637
anonymous proposals	0.785184	0.628641	0.393802
mismatched metadata	0.785092	0.629142	0.393495
shuffled node addresses	0.785063	0.629387	0.394104

Node declarations increase binary macro-F1 from 0.785010 to 0.785513, change multiclass macro-F1 from 0.630392 to 0.629835, and reduce regression NRMSE from 0.393695 to 0.393637. Semantic proposals exceed the anonymous and node-shuffled controls on both classification means; regression favors mismatched metadata over the semantic bank. These comparisons locate the effect of the node interface separately from the joint-training and revision comparisons in Tables 3 and 4.

G TRAINING BUDGET, KNOWLEDGE SAMPLING, AND ORDERING

G.1 OPTIMIZATION BUDGET AND INDEPENDENT TASK COVERAGE

Additional training can improve knowledge use through more optimization on familiar tasks or broader exposure to new tasks. We compare the reference reader after 2,048 joint updates with two continuations to 8,192 updates: replaying the original optimizer blocks and replacing the additional blocks with fresh synthetic tasks. Both continuations start from the same weights and optimizer state within each of three paired training seeds. The fresh tasks match the replay schedule's task type, support size, mechanism, information regime, class count, relation sign and strength, conflict type, and node-declaration mode. The architecture and prediction-plus-reconstruction objective remain fixed; node warm-up is not repeated.

All comparisons in this appendix use analytic reference predictions on a common panel of 72 worlds per task, with 128 query examples per world and paired class-prior settings. Regression's two prior views are identical and do not constitute independent repetitions. For multiclass prediction, $K = 3, 5, 10$ correspond to 128, 512, and 32 support examples, respectively; comparisons between class counts also change support size. This panel differs from the 24-world evaluation in the main text. We evaluate fixed final checkpoints and average task-specific scores over worlds, prior views, and three training seeds. These training variants are evaluated here on synthetic tasks; the real-task tables retain the original reader.

Table 13: Training-budget comparison. Entries are mean ± sample standard deviation across three training seeds, each evaluated on the same worlds. Parentheses give total joint updates. Generated knowledge follows the original missing/conflicting declaration process. F1 denotes macro-F1; lower NRMSE is better.

Knowledge	Training	Binary F1 ↑	Multi. F1 ↑	NRMSE ↓
Correct	Original (2,048)	0.68017 ± 0.00398	0.26945 ± 0.00597	0.38124 ± 0.00482
	Repeated (8,192)	0.69172 ± 0.00234	0.27383 ± 0.00669	0.36356 ± 0.00233
	Fresh (8,192)	0.69427 ± 0.00174	0.27618 ± 0.00110	0.36801 ± 0.00193
Generated	Original (2,048)	0.66802 ± 0.00202	0.25983 ± 0.00556	0.44187 ± 0.00392
	Repeated (8,192)	0.67239 ± 0.00226	0.26325 ± 0.00526	0.42574 ± 0.00256
	Fresh (8,192)	0.67596 ± 0.00039	0.26584 ± 0.00324	0.42829 ± 0.00237

Both continuations improve all three correct-knowledge metrics in every training seed (Table 13). Replaying the original data improves binary and multiclass F1 by 1.70% and 1.63% relative to the reference and reduces NRMSE by 4.64%. Fresh tasks yield corresponding improvements of 2.07%, 2.50%, and 3.47%. At the same 8,192-update budget, fresh exposure gives higher classification means, whereas replay gives lower regression error. The fresh-versus-replay classification gains occur in two of three binary seeds and one of three multiclass seeds. Thus, additional optimization provides the more consistent benefit in this comparison; independent task coverage has task-dependent effects.

G.2 KNOWLEDGE DISTRIBUTION AND CURRICULUM ORDER

To test how knowledge quality affects learning, we compare the original sampling distribution with a mixture of full-correct, sparse-correct, and locally conflicting declarations. Sparse views independently mask nodes and the reciprocal relation pair, including positive-only, negative-only, and mixed node subsets. Local conflicts alter one node declaration or one relation's sign or endpoint while retaining the other declarations. All groups use 2,048 joint updates, the same three paired initializations, and the same objective.

The new mixture contains 615 full-correct, 1,105 sparse-correct, and 328 local-conflict optimizer blocks. Each block contains all three task types. One arm randomly shuffles these blocks; the curriculum arm orders the identical blocks and knowledge-view seeds into stages of 410, 819, and 819 updates. Their approximate full/sparse/conflict proportions are 50/40/10, 30/55/15, and 20/60/20 percent. This pairing isolates order within the new distribution. The comparison with original sampling changes the knowledge inputs to prediction and reconstruction; the reconstruction corruption function itself is unchanged.

Table 14: Sampling and order comparison at 2,048 joint updates. Entries are mean ± sample standard deviation across three training seeds. The shuffled and staged arms share the complete exposure multiset. Each test view uses the same underlying worlds across training arms.

Knowledge	Training	Binary F1 ↑	Multi. F1 ↑	NRMSE ↓
Full correct	Original	0.68017 ± 0.00398	0.26945 ± 0.00597	0.38124 ± 0.00482
	Shuffled mixture	0.68897 ± 0.00691	0.26892 ± 0.00262	0.36508 ± 0.00469
	Staged order	0.68682 ± 0.00058	0.27036 ± 0.00542	0.36630 ± 0.00279
Sparse correct	Original	0.66189 ± 0.00152	0.24432 ± 0.00533	0.46868 ± 0.00798
	Shuffled mixture	0.67262 ± 0.00509	0.25537 ± 0.00759	0.44115 ± 0.00234
	Staged order	0.66861 ± 0.00066	0.25572 ± 0.00708	0.44263 ± 0.00247
Local conflict	Original	0.67858 ± 0.00122	0.26813 ± 0.00544	0.39079 ± 0.00282
	Shuffled mixture	0.68478 ± 0.00256	0.26768 ± 0.00180	0.38483 ± 0.00381
	Staged order	0.68092 ± 0.00367	0.26833 ± 0.00264	0.38269 ± 0.00348

The shuffled mixture improves sparse-knowledge prediction in all three tasks and all three training seeds (Table 14). Relative to original sampling, binary and multiclass F1 increase by 1.62% and 4.53%, and NRMSE decreases by 5.87%. Full-correct and local-conflict inputs also improve on binary and regression means, while multiclass F1 changes slightly downward. Increasing exposure to incomplete but correct knowledge therefore improves its use without requiring a staged order.

Ordering the same blocks does not produce a uniform additional gain. On sparse inputs, the staged arm slightly improves mean multiclass F1 over the shuffled arm, but lowers binary F1 and increases regression error. The multiclass increase occurs in one of three seeds. Local-conflict regression instead favors staged order. These differences distinguish the effect of the knowledge distribution from the effect of sequencing it.

G.3 KNOWLEDGE-ERROR CONTROLS AND SEMANTIC RECOVERY

We also evaluate the same trained readers under the original generated knowledge process, simultaneous node/pair-sign flips, and address shuffling (Table 15). The flip control retains the true relation addresses; shuffling changes where declarations attach. These interventions test whether gains on sparse or locally conflicting inputs extend to other forms of error.

Table 15: Error controls on the common synthetic panel. Entries average three training seeds. Node/pair flips reverse node declarations and pair signs; address shuffle permutes node and pair coordinates. Parentheses give total joint updates.

Knowledge	Training	Binary F1 ↑	Multi. F1 ↑	NRMSE ↓
Generated	Original (2,048)	0.66802	0.25983	0.44187
	Repeated (8,192)	0.67239	0.26325	0.42574
	Fresh (8,192)	0.67596	0.26584	0.42829
	Mixture (2,048)	0.66985	0.26168	0.44721
	Staged (2,048)	0.66762	0.26130	0.44257
Node/pair flips	Original (2,048)	0.66469	0.26251	0.40545
	Repeated (8,192)	0.66555	0.25981	0.38996
	Fresh (8,192)	0.66821	0.26242	0.39179
	Mixture (2,048)	0.66387	0.25719	0.40952
	Staged (2,048)	0.66239	0.25844	0.40483
Address shuffle	Original (2,048)	0.65274	0.25654	0.49309
	Repeated (8,192)	0.65041	0.25602	0.47898
	Fresh (8,192)	0.65281	0.25792	0.48260
	Mixture (2,048)	0.64284	0.24853	0.53400
	Staged (2,048)	0.64168	0.24965	0.52129

The new mixture's sparse-input improvements do not extend to arbitrary address corruption. Under shuffling, binary and multiclass F1 decrease from 0.65274/0.25654 to 0.64284/0.24853, and regression NRMSE increases from 0.49309 to 0.53399. On the original generated views, its regression error also increases from 0.44187 to 0.44721. Local-conflict exposure and broad coordinate corruption therefore impose different requirements on the reader.

Semantic recovery shows a related tradeoff. Across the full, sparse, and local-conflict views in all three tasks, the new shuffled mixture lowers node Brier error but increases signed-edge reconstruction MSE relative to original sampling. For example, under sparse binary knowledge, edge MSE changes from 0.15616 to 0.16350; for regression it changes from 0.06451 to 0.06672. In the local-conflict multiclass view, recovery of false-absent nodes rises from 17.59% to 23.15%, while rejection of false-present nodes falls from 97.22% to 91.67%. The larger recovery rate does not imply uniform error correction. Together, these comparisons identify incomplete correct knowledge as a useful training component and motivate retaining broader error controls when changing its sampling frequency.

H KNOWLEDGE UTILITY AND BOUNDED ADAPTIVE SAMPLING

Separating correctness from utility. Correct knowledge need not add predictive information when the reference predictor already explains the target. We therefore extend the knowledge-sampling comparison with fully explained reference predictions and broad address corruption. In the fully explained condition, both support and query reference predictions equal the generator's conditional mean or class probabilities; observations, sampled labels, and knowledge targets remain unchanged. Correct declarations then add no information for squared-error or log-loss prediction, although macro-F1 can still change through the decision rule. This analytic control tests unnecessary corrections rather than backbone accuracy.

Matched training distributions. The reference is the shuffled full/sparse/local-conflict mixture in Appendix G.2. Fixed coverage assigns probabilities 0.20 and 0.30 to complete and sparse knowledge; 0.05 each to false-present, false-absent, pair-sign, and single-endpoint errors; 0.10 to global address shuffling; and 0.10 each to complete and sparse knowledge with the fully explained reference. These nine conditions separate knowledge content from what remains unexplained by prediction. All three arms share the same three initializations, world–task–support schedule, optimizer, joint objective, and 2,048 updates. The expanded distribution changes knowledge views and includes fully explained references; it does not change the template or reader architecture.

The adaptive arm retains the fixed distribution as a coverage floor and reallocates at most 20% of sampling mass. Six disjoint training-probe worlds per task evaluate every condition every 256 updates. A condition becomes eligible when its conditional prediction risk decreases by more than 1% and 10^{-6} since the previous probe. Eligible weights are proportional to the fixed probability times current risk, clipped to $[0, 1]$; probabilities move halfway toward the resulting 80/20 mixture. Classification uses conditional KL divergence, and regression uses conditional-mean MSE divided by support-label population variance. Thus, high sampled-label error alone cannot trigger oversampling. Both new arms run the same monitoring probes, averaging approximately 249 seconds for optimization and 15 seconds for monitoring per replica on two CPU threads, excluding final evaluation.

Evaluation and selective-use gains. A fresh panel contains 72 worlds per task, paired bias views, and support sizes 32, 128, and 512; the regression bias views are identical. For multiclass tasks these support sizes coincide with $K = 3, 5, 10$, respectively. All knowledge conditions reuse the same worlds across three final-checkpoint replicas. Table 16 reports macro-F1 and support-sample-standard-deviation NRMSE; the weighted row uses the fixed probabilities above. These readers are evaluated as synthetic training controls, while the real-task tables retain the original reader.

Table 16: Coverage and adaptive-sampling controls on a new synthetic panel. Ref.: preceding shuffled mixture; Fixed: expanded coverage; Adapt.: bounded repair. Entries average three paired training replicas.

	Binary F1 ↑			Multiclass F1 ↑			NRMSE ↓		
Knowledge	Ref.	Fixed	Adapt.	Ref.	Fixed	Adapt.	Ref.	Fixed	Adapt.
Complete	0.69678	0.69075	0.69033	0.27374	0.27026	0.26772	0.36633	0.38266	0.38440
Sparse	0.68063	0.67358	0.67408	0.26412	0.25559	0.25557	0.42243	0.44310	0.44703
False-present	0.69393	0.68879	0.68839	0.27190	0.26959	0.26656	0.36702	0.38287	0.38480
False-absent	0.69642	0.69034	0.68967	0.27323	0.26942	0.26732	0.36973	0.38587	0.38832
Pair sign reversal	0.68839	0.68785	0.68746	0.27040	0.26749	0.26628	0.38801	0.39929	0.40009
One-endpoint error	0.67283	0.67047	0.67009	0.26278	0.26027	0.25839	0.49698	0.49229	0.49412
Address shuffle	0.65014	0.65894	0.65964	0.25659	0.25794	0.25634	0.52939	0.50210	0.50465
Explained, complete	0.72856	0.73000	0.73046	0.31186	0.31244	0.31157	0.24604	0.23441	0.23839
Explained, sparse	0.73010	0.72912	0.72978	0.31102	0.31113	0.31107	0.25021	0.23243	0.23664
Weighted mean	0.69200	0.68890	0.68906	0.27585	0.27222	0.27104	0.38365	0.38937	0.39232
Unknown	0.63566	0.63566	0.63566	0.21458	0.21458	0.21458	0.72419	0.72419	0.72419

Expanded coverage improves prediction under global address shuffling in every training replica: relative binary and multiclass F1 gains are 1.35% and 0.53%, and NRMSE decreases by 5.15%. With a fully explained reference, regression NRMSE decreases by 4.73% for complete and 7.11% for sparse declarations. Table 17 measures how much the reader perturbs an already correct conditional prediction: fixed coverage lowers this error in all six task–knowledge means, with regression reductions of 66.96% and 83.33%. These results show that exposing the reader to knowledge with different utility can reduce unnecessary corrections.

Table 17: Correction error when the analytic reference already equals the conditional target. Regression risk is conditional-mean MSE divided by support-label population variance. The unmodified reference has zero risk in all columns.

Knowledge	Sampling	Binary KL ↓	Multiclass KL ↓	Regression risk ↓
Explained, complete	Reference	0.01803	0.02275	0.00721
	Fixed	0.01646	0.02110	0.00238
	Adaptive	0.01482	0.01834	0.00420
Explained, sparse	Reference	0.01858	0.02134	0.00869
	Fixed	0.01443	0.01592	0.00145
	Adaptive	0.01408	0.01495	0.00343

Retention and adaptive reweighting. The selective-use gains come with reduced use of informative knowledge. Fixed coverage lowers both F1 means and increases NRMSE under complete and sparse declarations in every replica. Its weighted binary and multiclass F1 decrease by 0.45% and 1.32%, and NRMSE increases by 1.49%. Node reconstruction also reflects this tradeoff: binary node Brier error increases from 0.13484 to 0.16301 under complete knowledge, but decreases from 0.22811 to 0.22384 under shuffling. Adaptive repair further reduces classification KL error with fully explained references, yet raises the corresponding regression error and does not improve the weighted task metrics uniformly. The comparison therefore supports explicit evaluation of knowledge utility, while showing that broader coverage or competence-based reweighting alone does not ensure preservation of informative-knowledge gains.