

Zeno: Episode-Based Meta-Forecasting with Temporal and Relational Memory

Marco Barbosa

Zeno Divergent `zenodivergent.dev`

2 October 2026

Abstract

Forecast agreement does not by itself establish reliability: several forecasters can share a bias, an information source, or a failure mode. We specify the episode-based architecture being developed for Zeno, a model of forecasting behaviour, and distinguish it from the currently implemented warning prototype. The unit of learning is a dated forecasting episode containing a panel of forecasts, each forecaster’s available history, and supported relationships between their past errors. A temporal encoder represents changing forecaster behaviour; a relational set encoder represents the panel; and a baseline-anchored decoder produces one predictive distribution over the eventual outcome. Correction, interval-escape risk, and joint-failure risk are derived from that distribution. Training combines behavioural pretraining with episode-level probabilistic assessment, including sparse-history and unfamiliar-forecaster settings. We specify the information boundaries, representation, learning objectives, and evaluation protocol for the first Episode-Lite implementation. The current implementation and downloadable CPU beta are described separately. This manuscript does not report validated performance by the full episode architecture. Retrieval and evidence acquisition remain subsequent extensions.

Keywords: meta-forecasting; conditional reliability; shared errors; probabilistic ensembles; forecasting episodes

1 Introduction

An ensemble can become more certain without becoming more correct. Its members may use similar observations, inherit related assumptions, or repeat the same error under a changing regime. Conversely, disagreement can reflect useful complementary information. The practical question is therefore conditional: *given this forecasting situation, which expectations deserve confidence, and what failure remains plausible?*

Zeno treats a forecast as an observation of a forecasting system. Its inputs include the forecast itself, what happened to that system’s earlier forecasts, and how its errors relate to those of the other systems present. The model learns to assess a panel rather than assign an enduring reputation to each member. This permits the same architecture to represent a numerical weather ensemble, a health forecasting hub, an economic survey, or a set of event probabilities, while preserving their different targets and scoring rules.

The implementation target has three defining elements: (i) temporal states learned from dated forecaster experience, (ii) relational attention informed by matched past outcomes, and (iii) one predictive outcome distribution that supports multiple consistent assessments. The intended contribution is their integration within a reproducible, decision-time episode contract. Set learning and attention are established techniques [1, 2]; neither constitutes a novelty claim here.

The first complete architecture, *Episode-Lite*, targets 5–10 million parameters. The current prototype uses a 320-dimensional, six-layer set encoder, approximately 7.4 million parameters; its implemented inputs and outputs remain narrower than the architecture specified here. It will be compared with the existing Crowd-TransformerV3 warning model and simpler statistical

references. The previously released fragility beta is a separate estimator; its measurements are not results for this architecture.

2 The forecasting episode

2.1 Inputs, identities, and supervision

An episode is defined at assessment time t for a single realised outcome y_e :

$$\mathcal{E}_{e,t} = (q_e, \mathcal{F}_{e,t}, \mathcal{H}_{e,t}, R_{e,t}, C_{e,t}). \quad (1)$$

Here q_e specifies the target, units, horizon, bounds, and resolution rule; $\mathcal{F}$ is the available forecast panel; $\mathcal{H}$ contains the members’ resolved histories; R contains supported pairwise relationships; and C contains dated context and provenance. The outcome and its release time are held in a separate supervision block.

Three identities have different jobs. A *forecast ID* identifies a particular submission or revision. A *canonical outcome ID* joins forecasts to the same realised quantity and determines evaluation grouping. A *history-series ID* connects successive forecasts by a system for compatible targets, locations, units, and horizons. Including an outcome date in the history-series ID would fragment memory into unrelated events.

2.2 The information boundary

Every input must have been available by t . A past error enters memory only after its outcome became knowable. Source publication, local capture, forecast issue, target, and outcome release are distinct clocks. Retrospective reconstruction does not establish that a record was publicly accessible at its claimed issue time.

A label may legitimately arrive after the forecast being trained on. This differs from supplying that label

as an earlier input. At historical model-fit cutoff T , supervision must additionally satisfy $t_{\text{label}} \leq T$. Unresolved records can support input reconstruction tasks but cannot supply outcome losses.

Multiple forecasters, revisions, quantile bins, and thresholds associated with one outcome remain related evidence. They are not independent evaluation observations. For pair statistics, each member contributes one eligible forecast per historical outcome under a declared horizon and revision policy. Revisions remain available as sequences; they do not inflate pair support.

2.3 Representing the available corpus

Native forecast structure is preserved. Numeric distributions retain quantile levels or CDF grids, bounds, and tail conventions. Binary and categorical forecasts retain complete aligned probability vectors. Point forecasts remain point forecasts unless a residual distribution has been fitted using earlier data. Missing intervals do not imply zero uncertainty.

The episode interface is intended to cover weather, COVID-19, RSV, FluSight, ECB surveys, Forecast-Bench, and crypto. Eligibility is determined per episode. A source name is not proof that its records contain usable panels, histories, or outcomes. Single-member market episodes can teach calibration, but cannot identify an advantage from combining multiple forecasters.

Jev judgments enter only as typed, uncertain context with their actual availability times, evidence references, and rule versions. Later judgments are not backfilled into earlier episodes. Retrospective difficulty and coherence pilots remain diagnostic. No judgment becomes an outcome label, a causal fact, or an automatic training weight.

3 Architecture

3.1 Forecast and temporal encoders

A type-aware forecast encoder maps member i 's native forecast and target metadata to a vector x_i . Domain-dependent units are normalized using training-only assets. The representation retains missingness, forecast age, horizon, and the kind of uncertainty supplied. It distinguishes a forecaster's stated interval from a range constructed from disagreement.

The history encoder receives a sequence of resolved experiences:

$$H_i(t) = \{(r_{ij}, \ell_{ij}, v_{ij}, a_{ij}, m_{ij})\}_{j: a_{ij} \leq t}, \quad (2)$$

where r denotes a standardized residual, ℓ a forecast loss or coverage observation, v revision information, a the label availability time, and m masks and metadata. All normalizers and reference quantities used to construct these experiences must themselves be fitted without future labels.

The temporal state and member embedding are

$$h_i = \text{Hist}_\theta(H_i(t), \text{ages}, \text{masks}), \quad (3)$$

$$z_i = \phi_\theta(x_i, h_i, q_e). \quad (4)$$

Temporal attention encodes sequence order and elapsed time, allowing the same forecaster to have a different

state in different episodes. A learned missing-history representation and explicit support count handle cold starts. Stable forecaster IDs are essential for joins; the model should not rely on an arbitrary ID as a shortcut for permanent reliability.

Training varies available history length, includes entirely unfamiliar forecasters, and drops panel members to represent abstention. Revision sequences are restricted to revisions available at t . A model-version change is an input only when the source records it; it is not inferred as fact from a change in error.

3.2 Matched relational memory

For members i and j , let $\mathcal{O}_{ij}(t)$ be their distinct shared historical outcomes that were knowable before assessment. Relations summarize matched residual dependence, bias-adjusted co-movement, joint tail events, and support. A raw co-miss count alone is not a dependence estimate: frequent individual misses also produce frequent joint misses.

For a fixed miss definition, a useful diagnostic is

$$d_{ij} = \hat{p}_{ij} - \hat{p}_i \hat{p}_j, \quad \tilde{d}_{ij} = \frac{n_{ij}}{n_{ij} + \kappa} d_{ij}, \quad (5)$$

with all probabilities measured on the same matched outcomes and $n_{ij} = |\mathcal{O}_{ij}(t)|$. The shrinkage strength κ is fitted or selected using earlier data. Task difficulty and systematic bias must also be accounted for when interpreting dependence. Correlated errors do not identify a causal relationship.

Above-median absolute error for a point forecast, escape from an own 90% interval, and peer-relative confident failure are distinct events. Their definitions and support travel with the features. Unsupported pairs carry a mask; they are not encoded as measured independence. Provenance or Jev relationship probabilities augment this representation without replacing observed support.

3.3 Relational set attention

Panel processing is permutation equivariant before an invariant pooling operation [1, 2]. A relation-aware attention layer can use

$$\alpha_{ij} = \text{softmax}_j \left(\frac{Q(z_i)^\top K(z_j)}{\sqrt{d}} + b_\theta(R_{ij}) \right). \quad (6)$$

Here R_{ij} includes support and missingness as well as estimated dependence. An unknown relationship masks the relation-specific contribution; it does not prevent ordinary attention between two available forecasters. Padding masks exclude absent members. The episode representation is

$$u_e = \text{Pool}(\text{Set}_\theta(\{z_i\}, R, q_e)). \quad (7)$$

Global context is kept separate from member-specific features. Light target- or source-specific adapters can express different output semantics while sharing the behavioural representation. Their benefit must be compared with fully shared and domain-only alternatives.

Full pair attention costs $O(K^2)$ for a panel of K forecasters. Sparse observed-history edges reduce relation

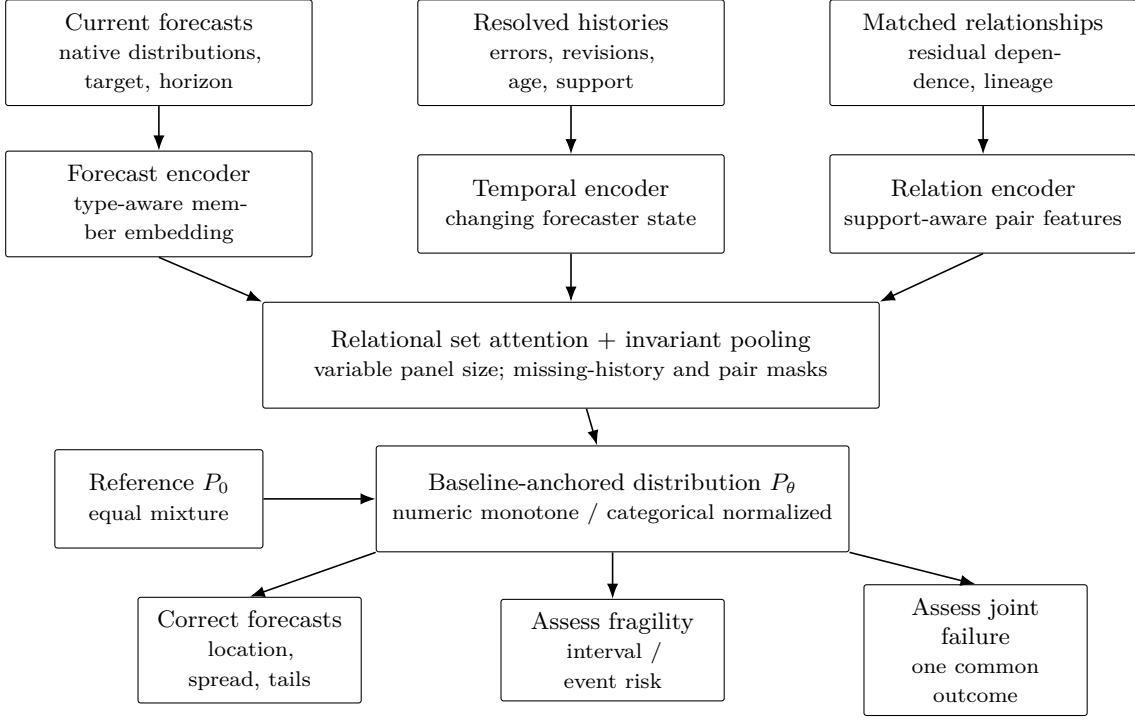


Figure 1: The Episode-Lite architecture under implementation. Forecaster order does not affect the episode prediction. Histories and relationships are reconstructed at the assessment time. The three output views are functions of one predicted outcome distribution, rather than independently fitted claims about mutually related events. Retrieval and evidence acquisition are outside the first implementation.

storage, but do not by themselves make dense attention linear. Large panels therefore require measured memory use, padding-aware batching, or an explicit approximation. Silent truncation to the first 64 members is not an acceptable panel policy.

3.4 A coherent predictive distribution

The model estimates $P_\theta(Y | \mathcal{E})$, not a collection of unconstrained warning scores. Its baseline P_0 is the equal mixture of the eligible member distributions. Numeric CDFs are aligned and mixed in distribution space; averaging quantiles is a different operation. Bounds, discrete masses, and open-tail conventions are fixed by the target adapter.

The decoder is anchored so that its initial prediction equals P_0 . For numeric outcomes, a monotone transformation of the reference variable provides one compatible specification:

$$X \sim P_0, \quad Y = T_\theta(X; u_e), \quad (8)$$

$$T_\theta(x; u) = m_0 + s_0 \left[b_\theta(u) + \int_0^{(x-m_0)/s_0} e^{g_\theta(v, u)} dv \right]. \quad (9)$$

At $b_\theta = 0$ and $g_\theta = 0$, $T_\theta(x) = x$. Positive derivatives preserve CDF ordering. Location shift and scale changes can move predictions outside the forecasters' original numeric range. The reference scale $s_0 > 0$ uses a historical residual-scale fallback when panel spread collapses. Transforming to a suitable target coordinate, or using an explicit bounded decoder, preserves legal support.

Equation (9) formalizes the decoder requirement; a monotone spline or another equivalent implementation

must satisfy the same identity, support, and gradient checks. Tail extrapolation and numerical integration are part of the executable specification, not left to serving-time defaults.

For categorical targets, residual logits give

$$p_\theta(c | \mathcal{E}) = \text{softmax}_c (\log p_0(c) + \delta_\theta(c, u_e)). \quad (10)$$

Zero residual logits retain the reference. Any probability floor is declared in the baseline policy and applied identically during fitting, evaluation, and serving. Binary forecasts are the two-category case. Ordered discrete outcomes retain their legal mass representation.

3.5 Correction, surprise, and joint failure

For member i , let $g_i(y; \mathcal{E})$ be a predeclared failure event. Its risk is

$$r_i = \mathbb{E}_{Y \sim P_\theta} [g_i(Y; \mathcal{E})]. \quad (11)$$

Examples include leaving an own interval, making a consequential directional error, or satisfying a precise confident-and-wrong rule. The panel event that at least m members fail has probability

$$r_{\geq m} = \mathbb{E}_{Y \sim P_\theta} \left[\mathbb{I} \left\{ \sum_i g_i(Y; \mathcal{E}) \geq m \right\} \right]. \quad (12)$$

One sampled outcome evaluates every forecast in the episode. This preserves logical relationships between event risks and distribution corrections; it does not guarantee calibration. The point prediction is the appropriate functional of the same distribution, such as its median under absolute-error loss. An outcome can be surprising without implying that a calibrated forecaster was defective.

4 Preparation and learning

4.1 From source records to episodes

The preparation pipeline verifies source identity and schema, preserves complete pagination, and freezes checksummed shards at a stated cutoff. It builds canonical outcome joins, histories, and pair features before optimization. Features are not updated by repeatedly replaying labels during training epochs. Missing values remain explicit.

Chronological training, development, and confirmation partitions are formed by canonical outcome groups. Revisions and cross-source copies of the same outcome stay together. Label maturity is checked at each model-fit cutoff, and boundary groups are purged or moved according to the frozen policy. An exposure ledger excludes previously inspected outcomes from claims of untouched confirmation.

The reader and feature builder must support an end-to-end replay: raw records, dated episode, tensors, prediction, and evaluation. Perturbing any future outcome must leave earlier inputs unchanged. Pair counts must reconcile to distinct matched outcomes on both sides of each pair. Sequence encoders require actual dated experience records; a history-summary count alone is not a temporal training sequence.

4.2 Stage 1: behavioural pretraining

Pretraining learns representations of forecasting behaviour using earlier experience as context and a later forecast as the query. Candidate tasks predict the next standardized error distribution, coverage, or supported co-failure event. Revision reconstruction uses only the revision prefix available at the query time. Unresolved records are restricted to input-only reconstruction objectives.

The pretraining objective is a masked sum,

$$\mathcal{L}_{\text{pre}} = \sum_a \lambda_a \mathbb{E} [M_a \ell_a(\hat{y}_a, y_a)], \quad (13)$$

where M_a indicates whether task a has a valid label. Task weights and scales are training-only assets. Missing pair labels cannot be treated as negative co-failure examples. Each auxiliary objective remains a hypothesis: it is retained only if it helps the final forecasting task under matched evaluation.

4.3 Stage 2: episode assessment

The pretrained encoders initialize the assessor; the residual output heads initialize at the reference. The main objective is a proper score for the predicted distribution [3]. Numeric targets use CRPS or a declared quantile/interval approximation; categorical targets use logarithmic or Brier loss. A quantile approximation is not silently equated with the full tournament log score.

With domain d , a typical objective is

$$\mathcal{L}_{\text{assess}} = \sum_d w_d E_{e|d} \left[\frac{S_d(P_\theta, y_e)}{s_d} \right] + \lambda_{\text{ref}} \Omega(P_\theta, P_0), \quad (14)$$

where s_d is a training-fitted loss scale, w_d is the declared sampling/weighting policy, and Ω penalizes unsupported

departure from the reference. Incompatible raw losses are not pooled without scale and weighting definitions. Final reports retain original-unit scores per target.

History masking, panel dropout, and varying panel sizes expose the model to sparse support. Small domains receive bounded reuse rather than unrestricted oversampling. Pretrained encoders and new output heads can use separate learning rates. These are controlled training choices, not substitutes for missing information. A rehearsal must verify that nonzero supervised errors reach the correction branch and, after initialization, the shared encoders.

Checkpointing preserves optimizer, scheduler, sampler, and random states. Learning-rate schedules are matched to the actual step budget. Selection uses development data and the exact serving transformation. Shared-gradient participation is an engineering diagnostic; it does not establish useful cross-domain transfer.

5 Evaluation protocol

5.1 Comparators and distinct questions

The principal comparison is the complete predictive distribution against P_0 , a calibrated equal mixture, simple bias/spread correction, and recent-performance weighting. A strong tabular model uses the same information. Domain references include appropriately specified volatility models for crypto and existing ensemble/calibration methods for weather and health. The CrowdTransformerV3 comparator is scored on matched warning labels; a warning-only model cannot by itself establish the quality of an outcome-distribution decoder.

Four claims are evaluated separately:

- **Forecast quality:** proper score, calibration, interval coverage and width, direction, and error severity.
- **Failure anticipation:** Brier/log loss for defined failure events, ranking, and alert precision/recall under a policy that can run prospectively.
- **Transfer:** domain-only versus joint training, unfamiliar forecasters, and chronological out-of-domain tests.
- **Intervention value:** improvement after an additional source or action, measured in a later experiment.

ROC AUC uses tie-correct ranks; a constant predictor must score 0.5 when both classes are present. It is secondary to proper probability scores. An alert rule must use a training-fitted threshold or a prespecified live batch budget, not the best 10% selected across a completed future period.

5.2 Ablations and uncertainty

The core ablations compare direct episodic training with behavioural pretraining, own-history inputs with matched relational memory, and eligible provenance/Jev context with its absence. Parameter count, information access, and compute are matched where required. A few-shot test withholds forecaster identity and varies the number of resolved support episodes.

Differences are computed on the same outcomes and resampled in outcome/time blocks appropriate to each

Table 1: Current prototype versus the architecture target.

Component	Status at the inspected revision
Panel attention	Implemented over member summaries.
Error pretraining	Implemented; scalar normalized error target.
Temporal memory	Ordered histories not consumed.
Pair relationships	Present in feeds; not consumed by the model.
Typed outputs	Current runner accepts quantile panels only.
Outcome decoder	Not present in the warning prototype.
Jev context	Not consumed by the current runner.

source. Thousands of forecaster rows around two economic outcomes remain two outcome groups. Small panels are reported descriptively when they cannot support stable inference. Seed variability and uncertainty from sampled outcomes are reported separately.

All model, calibration, and per-source deployment choices are frozen before one confirmation evaluation. Repeated selection on development data is acknowledged as optimistic. Passing a confirmation test does not erase limited seasonal coverage. A source can remain on the reference method even when another source supports a learned adjustment.

5.3 Training telemetry

Each run records the manifest and code hashes, unique outcome counts, history and pair support, label eligibility, optimizer exposure by source, native-unit and normalized losses, baseline skill, calibration, per-source gradients, learning rate, time, and compute cost. Selected and latest checkpoints are shown separately. The public training view reads the same versioned events as the run record rather than a copied headline.

6 Implementation scope and limitations

This manuscript separates an architectural specification from its current implementation. The repository snapshot inspected on 2 October 2026 contains an `EpisodeLite` prototype in `model/v0393/run.py` [4]. It maps 11 summary features per member into a six-layer set Transformer, with a source embedding, a scalar error head, and shared plus source-specific warning heads. The 4.8M comparator has the same general structure at a smaller width. The current job invokes this training entry point; the private launch archive was not available for independent byte-for-byte comparison.

The prototype has supervised error pretraining, but no temporal sequence encoder. Its feature builder does not consume ordered experiences, pair dependence, lineage, or Jev context. It requires three numeric quantiles and therefore rejects the live ECB and ForecastBench panels whose uncertainty is not represented in that format. It emits warning logits rather than the outcome distribution in Section 3. The complete architecture in Figure 1 is the implementation target, not a description of capabilities already demonstrated by that job.

The first release target is one frozen checkpoint, its preprocessing assets, the episode contract, and an ex-

ecutable example. Python, API, and MCP interfaces should return the same candidate distribution, reference, support, limitations, and model identity. Unsupported inputs produce an explicit refusal or reference fallback. Offdiagonal preserves evidence, Zeno assesses it, Zeno30 records agent forecasts, and the Arena scores frozen candidates beside the deployed policy on the same episodes.

Data diversity is useful only where clocks, joins, and target semantics survive preparation. Short weather history, repeated economic outcomes, reconstructed crypto methods, unavailable lineage, and sparse pair overlap constrain what the first model can learn. Distinct people or teams are not necessarily statistically independent. Conversely, a shared provider is not proof of identical errors.

Episode retrieval and an intervention selector are subsequent extensions. A withheld-forecaster study can test which additional forecast would have helped, with hidden forecasts and their derived metadata removed from inputs. It cannot establish the value of an autonomous research action without prospective, equal-budget comparisons. Larger parameter budgets are capacity hypotheses rather than evidence of stronger forecasting.

7 The available Fragility Beta

The public download, Zeno Fragility Beta, is a separate CPU research artifact [6]. It contains per-source logistic and gradient-boosted estimators over six features: crowd distance, long and recent track record, crowd spread, crowd size, and own confidence. Its configuration lists 38,459 fitting rows across six warning packages and 62,880 stored learned numbers. These are not the training population or parameter count of the new neural model. The advertised upstream corpus size describes a different stage of the data funnel.

The current Python helper now chooses the source’s selected estimator. However, the model card’s market ONNX example points to the tree estimator while Python selects logistic regression. For the card’s Python feature example, the downloaded logistic model returns 0.1921 and the tree model returns 0.03915. These are reproducibility diagnostics, not measured failure frequencies. The deployed market logistic coefficients are effectively nonzero only for own confidence, so its incremental value must be tested against an equivalently oriented confidence baseline.

The published reports contain earlier development ranking results; correcting the AUC implementation in source code does not itself regenerate them. AUC and top-budget lift do not establish calibrated failure probabilities or earlier warnings. The release also requires a common estimator identity across its reports, refitted artifacts, Python, ONNX, and service examples. Accordingly, this paper identifies the beta as downloadable and executable while treating its probability, lead-time, and incremental-skill claims as requiring reconciliation. It is not evidence that temporal memory, shared-error reasoning, or cross-domain transfer has been validated.

8 Conclusion

Zeno’s episode architecture models changing forecaster behaviour and relationships in the context of one future outcome. Its central constraint is that correction and failure assessment agree with the same predictive distribution. Behavioural pretraining, matched relational memory, and explicit information boundaries provide a concrete route to testing whether a model of forecasters adds value beyond averaging and calibration. The next empirical question is whether these components improve decisions on genuinely later episodes, not whether the architecture can fit a larger corpus.

Reproducibility and status

Architecture manuscript, revision 3; not peer reviewed. No performance result is attributed here to Episode-Lite. Design scope follows the current implementation plan and source snapshot [4, 5]. The separate existing beta is identified by the pinned revision in reference [6]. Trained weights, exact tensor dimensions, target adapters, and run metrics require a subsequent versioned model report.

References

- [1] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola. Deep Sets. In *Advances in Neural Information Processing Systems 30*, 2017. Paper.
- [2] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *ICML*, PMLR 97:3744–3753, 2019. Paper.
- [3] T. Gneiting and A. E. Raftery. Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi:10.1198/016214506000001437.
- [4] Zeno Divergent. Implementation source, 2 October 2026. Repository `mbarbosa30/zenodivergent`, revision `57a452fe05915`. Pinned source; episode reader.
- [5] Offdiagonal. Episode rules, 2 October 2026. Repository `mbarbosa30/flatplanet`, revision `ad00a1770ba78`. Pinned source. Source inspection does not certify all admission, clock, or pair-support checks.
- [6] Zeno Divergent. Zeno Fragility Beta. Hugging Face model repository, revision `dd881ddb0fbd`, inspected 2 October 2026. Pinned release. CPU research artifact, distinct from Episode-Lite.