TABBY-PRETRAIN: A RECIPE FOR CONTINUABLE PRETRAINING OF TIME SERIES FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretraining a time series foundation model is costly, making it desirable for a training run to remain effective as it is extended. In this paper, we develop a pretraining recipe that supports continuable forecasting improvements across successive pretraining stages through the coordinated design of optimization, supervision, and data, and instantiate it in TABBY-PRETRAIN, an encoder-only Transformer-based TSFM. The resulting recipe combines a progressive convergence schedule (PCS), which repeatedly produces annealed checkpoints; deep quantile supervision (DQS), which directly supervises intermediate layers; and CauKerV2, an online generator that composes diverse temporal dynamics through randomly sampled structural causal models. Controlled experiments with the 40M-parameter TABBY-SMALL show that replacing PCS, removing DQS, or replacing CauKerV2 with additional real data degrades forecasting at the same training budget. We then apply the recipe to a 146M-parameter model. Its reported training trajectory improves across three stable–decay stages, and the final pretrained backbone is competitive in forecasting while supporting frozen representation transfer to classification and anomaly detection.

1 INTRODUCTION

Time series are widely used in healthcare (Burger et al., 2024), finance (Xie et al., 2026b), transportation (Yanes-Pulido & Rodrigues, 2026), telecommunications (Zanouda et al., 2024), and many other domains (Wang et al., 2026). Despite their different origins, time series often share temporal structures such as trends, seasonality, local dependencies, and abrupt changes (Liu et al., 2026d; Kottapalli et al., 2025). This common structure makes it possible to pretrain time series foundation models (TSFMs) that transfer across datasets and domains (Xie et al., 2026c; Liang et al., 2024).

However, pretraining is costly, making it desirable for a training run to remain effective as it is extended rather than being tied to a single predetermined endpoint. We view continuable pretraining as this broader goal: successive stages should produce useful checkpoints while continuing to improve forecasting. Achieving this requires more than simply resuming optimization. Optimization must be organized so that later stages can proceed from converged checkpoints; predictive supervision should continue to shape useful transformations throughout model depth; and the training distribution should provide sufficiently diverse temporal dynamics as training proceeds. This leads to the central question of this study: *how can optimization, supervision, and data be coordinated so that a TSFM continues to improve across successive pretraining stages?*

We isolate this question from architectural novelty by using a patch Transformer (Nie et al., 2023; Wen et al., 2026), and address it through three aspects of pretraining. For optimization, long runs need annealed checkpoints from which learning can continue without overly aggressive restarts. Our progressive convergence schedule (PCS) uses repeated stable–decay stages with decreasing learning-rate peaks. For supervision, a terminal loss may leave intermediate blocks weakly involved in prediction. Deep quantile supervision (DQS) directly trains forecasts at selected depths and regularizes their prediction trajectories. For data, we introduce an online data generator CauKerV2 to complement real series.

We test these choices through fixed-step component ablations with the 40M-parameter TABBY-SMALL and targeted analyses of restarts and intermediate blocks. We then apply the recipe to the 146M-parameter TABBY-PRETRAIN model and evaluate the resulting backbone across multiple

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.

time series tasks. The model is pretrained with masked reconstruction under a quantile forecasting objective, and the same backbone is subsequently reused without encoder adaptation for classification and anomaly detection (Feofanov et al., 2026; Goswami et al., 2024). We report its three-stage training trajectory, zero-shot forecasting performance, and frozen-representation transfer results. Our contributions are fourfold:

- **PCS:** We organize continuable pretraining into repeated stable–decay stages with decreasing learning-rate peaks, producing annealed checkpoints for evaluation and continuation.
- **DQS:** We directly supervise quantile predictions at intermediate depths. Forecasting ablations and block-level analyses examine its effect on predictive performance.
- **CauKerV2:** We introduce an online generator that composes diverse temporal dynamics through structural causal models.
- **Full pretraining and open release:** We apply the recipe to TABBY-PRETRAIN. The pretrained backbone achieves competitive forecasting and provides frozen representations for classification and anomaly detection. To our knowledge, TABBY-PRETRAIN is among the few high performing forecasting TSFMs to release its complete pretraining pipeline alongside the model weights.

2 PRELIMINARIES AND PRETRAINING SETUP

Forecasting Objective and Plain Backbone. Let $\mathbf{x}_{1:T} = (x_1, \ldots, x_T) \in \mathbb{R}^T$ denote a univariate context and $\mathbf{y}_{1:H} = (x_{T+1}, \ldots, x_{T+H}) \in \mathbb{R}^H$ its future. A forecasting model f_θ maps the context to predictions for the next H values. To represent uncertainty, the model estimates conditional quantiles for a fixed grid $\mathcal{Q} \subset (0, 1)$. For horizon step h and quantile level τ, it predicts $\widehat{q}_{h,\tau}$ and minimizes the pinball loss

$$\mathcal{L}_{\mathrm{QT}} = \frac{1}{H|\mathcal{Q}|} \sum_{h=1}^{H} \sum_{\tau \in \mathcal{Q}} \rho_\tau(y_h - \widehat{q}_{h,\tau}), \qquad \rho_\tau(u) = u\big(\tau - \mathbb{I}[u < 0]\big).$$

TABBY-PRETRAIN is an encoder-only, non-autoregressive patch Transformer. During pretraining, forecasting is formulated as masked reconstruction within a fixed-length window containing both observed history and masked target positions. After mask-aware reversible normalization, the values are divided into non-overlapping patches, projected to token embeddings, and processed by bidirectional Transformer blocks. Masked future values are never exposed to the encoder. A shared decoder maps patch representations to monotone quantile forecasts, which are transformed back to the original scale. The full model supports windows of 8,192 observations; this is the model window rather than the available observed history for every sample. Channels of a multivariate series are processed independently. Complete equations and hyperparameters are given in Appendix A.

Masking and Base Data Sources. Following Contiguous Patch Masking (CPM) (Auer et al., 2025), we mask contiguous patch spans and optimize only the masked positions. Internal spans can use observations on both sides, whereas terminal spans require extrapolation from the preceding context and are kept short. A fixed-budget comparison of terminal-mask settings is reported in Appendix C.1.

The real-world data component combines GIFT-Eval-Pretrain+ (Aksu et al., 2024) and BLAST (Shao et al., 2025). GIFT-Eval-Pretrain+ extends the original collection with eligible complementary datasets (Auer et al., 2025; Podest et al., 2026) and uses sub-dataset-balanced sampling so that very large sources do not dominate. Closely related small datasets are consolidated to avoid disproportionate oversampling. BLAST is streamed directly because it already provides a strongly balanced distribution of temporal patterns, although its records are limited to 4,096 time steps. Full inventories, filtering rules, and corpus statistics appear in Appendix B.

The base synthetic source is KernelSynth (Ansari et al., 2024), which samples Gaussian processes from randomly composed kernels. To make long context generation practical, sequences are produced on a grid four times coarser than the target resolution and then interpolated; independent shards are generated in parallel. Implementation details are given in Appendix D.1. The full-model sampler assigns 30% each to GIFT-Eval-Pretrain+, BLAST, and KernelSynth, and 10% to CauKerV2. TABBY-SMALL uses the closely related mixture reported in Appendix C.

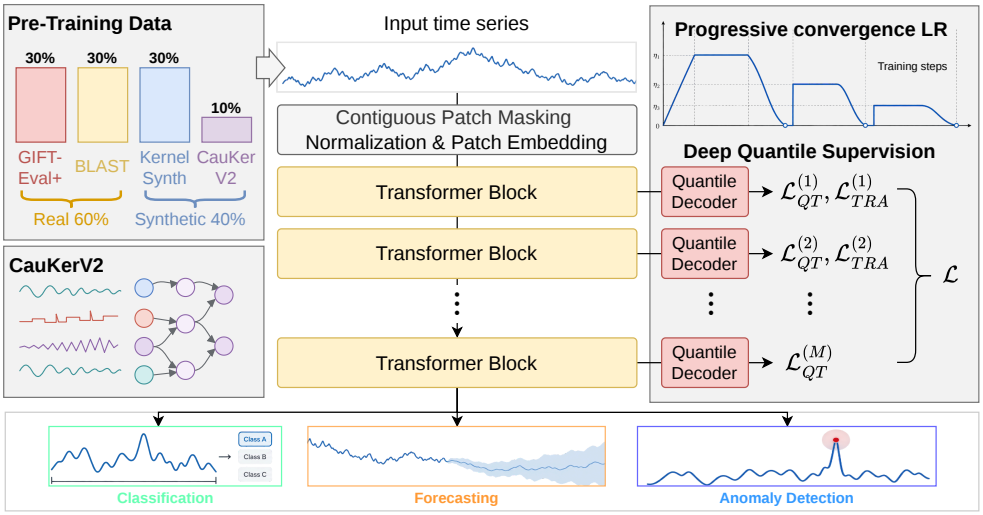


Figure 1: Overview of the pretraining recipe and downstream tasks.

3 A Recipe for Effective Continuable TSFM Pretraining

With the backbone and masking objective fixed, we design the pretraining recipe along three axes. PCS organizes continuation across stage boundaries, DQS changes how prediction signals reach intermediate depth, and CauKerV2 expands the temporal dynamics encountered by the model.

3.1 Progressive Convergence Schedule

Continuable pretraining requires both stage-end models that can be evaluated and starting points that can resume learning. Conventional cosine schedules are tied to a predefined endpoint, leaving intermediate checkpoints only partially annealed. Inspired by Warmup–Stable–Decay (WSD) and WSD-S (Hu et al., 2024; Wen et al., 2024), PCS uses a single warmup followed by repeated stable–decay stages, as shown in upper-right part of figure 1. For stage k, the stable learning rate η_k is

$$\eta_k = \eta_1 2^{-(k-1)}.$$

Thus, each stage operates at half the stable learning rate of its predecessor and ends with a decay phase that produces a locally annealed checkpoint. These checkpoints provide explicit intervention points from which training can be evaluated, reconfigured, or continued. The motivation for attenuating later restart peaks is examined in Section 4.3, and the complete full-model trajectory is reported in Section 5.1.

3.2 Deep Quantile Supervision

The schedule specifies how parameters are updated over time; DQS changes which learning signals reach the network at intermediate depth. Supervising only the terminal output may leave some blocks weakly involved in the predictive transformation (Wiliński et al., 2025; Bao et al., 2026). We therefore attach the shared quantile decoder to exits $\mathcal{E}$ of a model with M Transformer blocks. Here, $\mathcal{E}$ contains the initial exit 0 (before the first block), selected intermediate exits, and the terminal exit M. Let $\widehat{\mathbf{q}}^{(\ell)}$ be the quantile prediction at exit ℓ, and let $\mathcal{L}_{\mathrm{QT}}^{(\ell)}$ be the masked quantile loss defined in Section 2, evaluated using $\widehat{\mathbf{q}}^{(\ell)}$. Intermediate exits receive the auxiliary objective

$$\mathcal{L}_{\mathrm{DS}} = \sum_{\ell \in \mathcal{E} \setminus \{0, M\}} \left(\frac{\ell}{M}\right)^{\gamma} \mathcal{L}_{\mathrm{QT}}^{(\ell)},$$

where γ controls the weighting of losses across depth.

Intermediate quantile losses supervise each exit against the observed targets, but do not explicitly constrain how predictions evolve across depth (Lee et al., 2014; Abdessalem et al., 2026). We

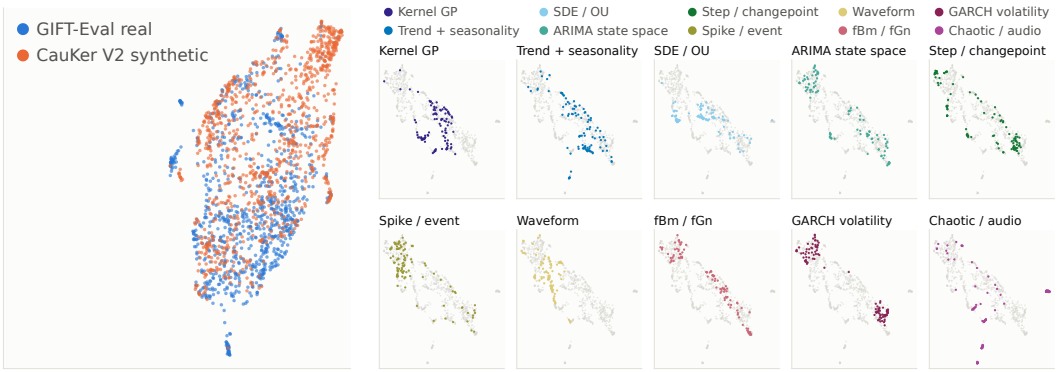


Figure 2: Embedding space view of the CauKerV2 training distribution.

therefore add a trajectory regularizer that encourages intermediate predictions to follow a simple interpolation from the initial prediction to the terminal prediction. This interpolation provides a depth-dependent reference rather than an additional ground-truth target:

$$\widetilde{\mathbf{q}}^{(\ell)} = \left(1 - \frac{\ell}{M}\right) \mathrm{sg}\left(\widehat{\mathbf{q}}^{(0)}\right) + \frac{\ell}{M} \mathrm{sg}\left(\widehat{\mathbf{q}}^{(M)}\right),$$

$$\mathcal{L}_{\mathrm{TRA}} = \sum_{\ell \in \mathcal{E} \setminus \{0, M\}} \mathrm{MSE}_{\alpha}\left(\widehat{\mathbf{q}}^{(\ell)}, \widetilde{\mathbf{q}}^{(\ell)}\right), \qquad \mathcal{L} = \mathcal{L}_{\mathrm{QT}}^{(M)} + \lambda_{\mathrm{DS}} \mathcal{L}_{\mathrm{DS}} + \lambda_{\mathrm{TRA}} \mathcal{L}_{\mathrm{TRA}}.$$

Here, sg denotes stop-gradient: its argument is used in the forward computation but treated as constant during backpropagation. MSE_{α} averages squared differences over masked positions and quantile levels, then weights each series by n_i^{α}, where n_i is its number of masked positions and $\alpha = 1/2$. The quantile loss uses the same mask and square-root length weighting. Finally, λ_{DS} and λ_{TRA} control the contributions of the two auxiliary losses. We evaluate the composite DQS objective through performance ablation, layer removal, and a local direction intervention in Section 4.2.

3.3 CauKerV2: Complementary Synthetic Temporal Dynamics

Effective continued training also depends on the distribution of training signals. CauKerV2 extends CauKer (Xie et al., 2026a) to provide an online synthetic source alongside real-world data and KernelSynth. While KernelSynth supplies Gaussian process series with composed kernels, CauKerV2 combines a broader portfolio of primitive dynamics through randomly sampled structural causal models (SCMs; Pearl (2009); Hollmann et al. (2023); Taga et al. (2025)).

The portfolio contains ten generator families, grouped here by their dominant temporal characteristics. Statistically structured or recurrent series come from kernel Gaussian processes, trend–seasonality decomposition, ARIMA/SARIMA state-space processes, regime-switching Ornstein–Uhlenbeck SDEs, and analytic waveform mixtures. Piecewise level/change-point and spike/event processes provide structural shifts and localized events. Fractional Brownian motion/noise, GARCH-family volatility processes, and chaotic or audio-inspired processes provide long-range dependence, volatility clustering, and nonlinear dynamics, respectively. Their definitions, sampling weights, and parameter distributions are given in Appendix D.2. Each primitive is assigned to a root node, and every non-root node is generated as

$$z_j(t) = \phi_j\left(\sum_{i \in \mathrm{Pa}(j)} w_{ji} z_i(t)\right) + \epsilon_j(t),$$

where $\mathrm{Pa}(j)$ is the parent set, w_{ji} is a sampled edge weight, and $\epsilon_j(t)$ is additive noise. The nonlinearity ϕ_j is sampled from linear, ReLU, leaky-ReLU, sigmoid, sine, and modulo transformations. Although the SCM produces multivariate samples, each output channel is presented to the channel-independent backbone as a separate univariate example. Online generation continually refreshes the synthetic stream and reduces repeated exposure to a fixed set of samples.

4 COMPONENT ABLATIONS AND DIAGNOSTICS

We compare PCS, DQS, and CauKerV2 under the same number of training steps. For DQS, we examine both the redundancy of intermediate blocks and whether its local update provides a benefit beyond increasing gradient magnitude. For PCS, we measure sensitivity to learning rate restarts at successive checkpoints. For CauKerV2, we visualize embeddings of generated and real time series.

4.1 EXPERIMENTAL SETUP AND COMPONENT CONTRIBUTIONS

We use TABBY-SMALL for controlled component analysis while holding architecture and training budget fixed. It contains 40.291M parameters, 12 Transformer blocks, and is trained for 110,000 steps over approximately three days on eight H200 GPUs. The data mixture and decoded exits differ from those of the full model and are reported explicitly in Appendix C. Unless stated otherwise, each comparison changes only the component under investigation. Each intervention replaces or removes one component from the complete recipe.

Table 1: Component comparisons on GIFT-Eval.

Setting	MASE ↓	CRPS ↓
PCS → Cosine	0.7456	0.5112
PCS → WSD-S	0.7551	0.5165
Remove DQS	0.7476	0.5117
Replace CauKerV2 with real data	0.7536	0.5109
TABBY-SMALL recipe	**0.7368**	**0.5038**

At the matched training budget, the complete TABBY-SMALL recipe reaches 0.7368 MASE and 0.5038 CRPS, lower than the Cosine (Wen et al., 2026) and WSD-S (Wen et al., 2024) schedule variants in Table 1. Removing DQS raises MASE by 1.47% and CRPS by 1.57% relative to the complete recipe. Replacing CauKerV2's 10% sampling share with GIFT-Eval-Pretrain+ raises them by 2.29% and 1.42%, respectively. These deletion and replacement comparisons support each component within the complete recipe at this fixed step budget; they do not establish an ordering of additions, interaction synergy, or optimality at equal FLOPs.

4.2 FUNCTIONAL AND OPTIMIZATION EFFECTS OF DQS

Redundancy Analysis. Prior studies have identified representational redundancy and limited sensitivity to intermediate-layer removal in several TSFMs (Wiliński et al., 2025; Bao et al., 2026). Motivated by these findings, we investigate whether DQS alleviates functional redundancy by directly supervising intermediate predictions. Table 2 presents the results for layers 6 to 8, using the zero-based indexing of Appendix E.1. Each relative change is measured against the corresponding checkpoint with all blocks retained.

Table 2: Functional ablation of intermediate blocks on GIFT-Eval. Negative values indicate improvements.

	ΔMASE (%)		ΔCRPS (%)	
Skipped layer	w/ DQS	w/o DQS	w/ DQS	w/o DQS
6	+1.13	+0.11	+0.91	-0.05
7	+1.52	+0.09	+1.07	-0.14
8	+1.63	+0.93	+1.45	+0.92

Without DQS, bypassing layers 6 and 7 has little effect on aggregate MASE and slightly improves CRPS. With DQS, bypassing either block increases MASE by 1.13%–1.52%. These results suggest that DQS encourages useful transformations at intermediate depth and reduces local functional redundancy in the tested checkpoints. This interpretation concerns aggregate sensitivity to single-block removal; it does not imply that blocks with small aggregate effects are irrelevant to every dataset. Full experimental settings and layer-wise results are provided in Appendix E.

Local update direction. To understand why DQS improves forecasting, we test whether it produces a more favorable update direction. At the fixed 110K TABBY-SMALL checkpoint, with the shared prediction head frozen and trajectory regularization disabled, we construct three one-step branches from the same parameters, AdamW state, update batch, and mask: final supervision only (A), a norm-matched rescaling of the final loss gradient (M), and final loss plus intermediate quantile supervision (B). Branch M matches B's raw gradient norm while preserving the direction of A. We apply the same optimizer state and clipping rule, rescale each parameter update to ρ times the norm of A's original update ($\rho \in \{0.25, 0.5, 1\}$), and evaluate terminal loss on held-out windows.

At the fixed 110K checkpoint, the DQS branch B yields lower held-out terminal pinball loss than both the final-only branch A and the raw-gradient-norm-matched branch M after controlling the actual parameter-displacement radius (Figure 3). Adding intermediate supervision increases the raw gradient norm by approximately 43.2%, yet A and M show no detectable loss difference in this probe. The cosine similarity between final and DQS gradients is 0.717; after AdamW preconditioning, B also has better update–validation-gradient alignment than M. These results support a locally more favorable update direction beyond gradient magnitude. They do not show that every DQS step reduces held-out loss or isolate the trajectory regularizer. Full construction and diagnostics appear in Appendix G.

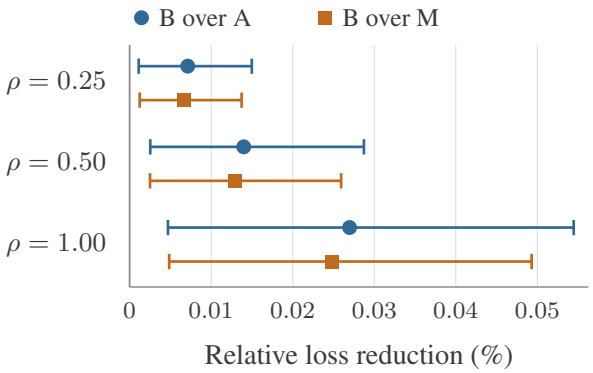


Figure 3: Optimizer-aware one step DQS intervention. Markers are macro-average relative reductions in final validation pinball loss and whiskers are 95% cluster-bootstrap intervals; positive values favor B.

4.3 Sensitivity to Aggressive Restarts

We further examine why PCS attenuates the peak learning rate at later stages. From the 55K and 110K TABBY-SMALL checkpoints and the 55K, 110K, and 165K full-model checkpoints, we restore both model and AdamW states and continue with $\eta_{\text{peak}} \in \{10^{-5}, 2.5\times 10^{-5}, 5\times 10^{-5}, 10^{-4}, 2\times 10^{-4}\}$. Each branch uses 1K constant-rate updates followed by a 500 step cosine cooldown to 10^{-5} and is evaluated on the same fixed pretraining-distribution validation probe.

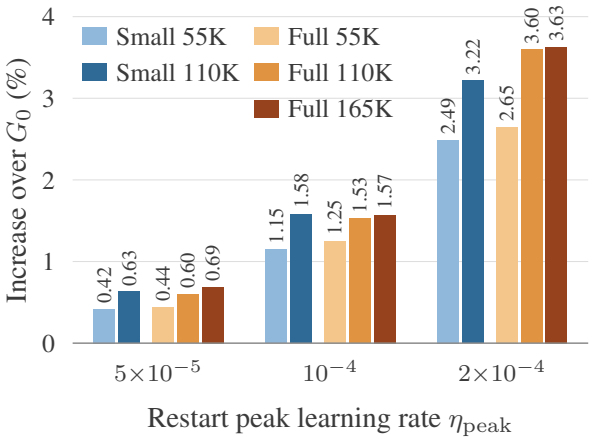


Figure 4: Sensitivity to restart peak learning rates. Bars give the relative increase in validation loss over the baseline G_0 measured at $\eta_{\text{peak}} = 10^{-5}$. Lower is better.

Figure 4 shows that the short-horizon cost of aggressive restart peaks increases from 55K to 110K at both scales and then largely saturates for the reported full-model checkpoints. This diagnostic includes full-model checkpoints even though the principal controlled ablations use TABBY-SMALL. It motivates smaller peaks at later stages. Complete settings and source-wise results are provided in Appendix H.

4.4 Embedding Space View of CauKerV2

To describe what this generator portfolio contributes, we encode series with the frozen Mantis encoder (Feofanov et al., 2026), reduce the resulting representations to ten principal components, and project them to two dimensions with UMAP (McInnes et al., 2020); the protocol is given in Appendix D.3. The left part of Figure 2 compares CauKerV2 samples with real samples drawn uniformly across GIFT-Eval-Pretrain sub-datasets. The two sets share the main body of the embedding, indicating that CauKerV2 produces series whose encoded characteristics also occur in real data, while each set retains regions that the other populates only sparsely. We read these projections as descriptive evidence of complementarity in a frozen representation space. The right part of Figure 2 shows that the ten primitive families occupy distinct but partially overlapping regions of this embedding. Smooth and statistically regular families, such as the kernel Gaussian process, trend–seasonality, and waveform, populate the interior of the pooled cloud, whereas families with abrupt structure, heavy tails, or long-range dependence, such as GARCH volatility, concentrate in separated peripheral modes. No single family covers the area spanned by the portfolio.

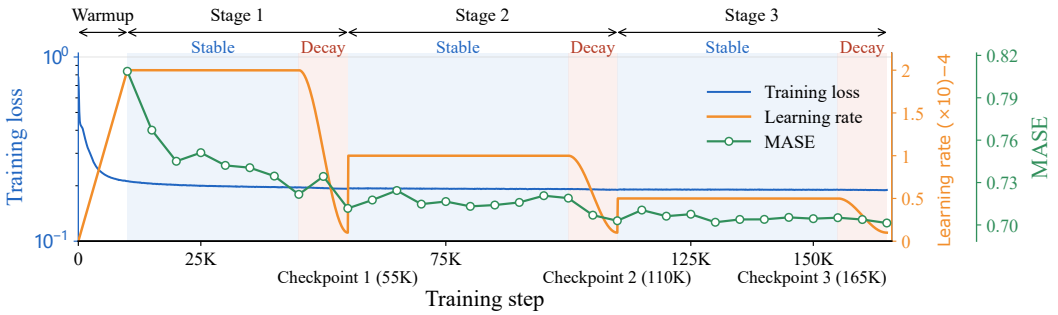


Figure 5: Continued pretraining of the full TABBY-PRETRAIN model over three stable–decay stages. MASE is evaluated on GIFT-Eval. Each decay produces an annealed stage-end checkpoint at 55K, 110K, and 165K steps.

5 FROM TABBY-SMALL TO TABBY-PRETRAIN

The controlled experiments identify component contributions at a fixed smaller scale. We now report the existing application of the same design principles to TABBY-PRETRAIN, which contains 145.843M parameters, 20 Transformer blocks, and hidden width 768. The complete 165K-step pretraining run took approximately six days on eight H200 GPUs. The full and Small configurations are not identical beyond parameter count; their data mixture, exits, and schedules are listed in the corresponding appendices.

5.1 CONTINUABLE PRETRAINING OF THE FULL MODEL

The full run follows PCS through three stable–decay stages. Figure 5 reports the complete trajectory. At 55K and 110K steps, the learning rate rises from a decayed endpoint to the lower stable rate of the next stage. On the reported curves, neither transition produces a visible large loss spike. Within each stage, the stable phase makes gradual progress and the decay yields an additional reduction before reaching a new plateau. The 55K, 110K, and 165K stage-end checkpoints are therefore usable evaluation points, while later stages continue to improve forecasting in GIFT-Eval benchmark. This single run illustrates the intended continuable pretraining behavior.

5.2 FORECASTING EVALUATION

Benchmarks and metrics. We evaluate the pretrained backbone on GIFT-Eval (Aksu et al., 2024), which contains 23 datasets across seven domains and ten frequencies, and TIME (Qiao et al., 2026), which contains 98 tasks from 50 recently collected datasets. We report Mean Absolute Scaled Error (MASE) for point forecasts and the discrete-quantile approximation of Continuous Ranked Probability Score (CRPS) for probabilistic forecasts:

$$\mathrm{MASE} = \frac{H^{-1}\sum_{h=1}^{H}|y_{T+h} - \widehat{y}_{T+h}|}{(T-m)^{-1}\sum_{t=m+1}^{T}|x_t - x_{t-m}|}, \qquad \mathrm{CRPS} \approx \frac{2}{H|\mathcal{Q}|}\sum_{h=1}^{H}\sum_{\tau\in\mathcal{Q}}\rho_\tau\left(y_{T+h} - \widehat{q}^{(\tau)}_{T+h}\right).$$

The median is used for $\widehat{y}$. Lower values are better. We interpret these results as benchmark forecasting performance without task-specific fine-tuning.

Main results. As shown in Figure 6, TABBY-PRETRAIN obtains MASE/CRPS of 0.7040/0.4827 on GIFT-Eval and 0.6779/0.5664 on TIME. On the official leaderboard, it outperforms Chronos-2-Synth (Ansari et al., 2025), Moirai-2 (Liu et al., 2026a), Sundial (Liu et al., 2026b), TimesFM 2.0 (Das et al., 2024), and Chronos-Bolt (Ansari et al., 2024) on both metrics of both benchmarks. On GIFT-Eval, it also outperforms PatchTST-FM-r1 (Wen et al., 2026), TiRex (Auer et al., 2025), and Toto 2.0 (Khwaja et al., 2026) on both metrics, achieving the lowest CRPS among the models shown. On TIME, it exceeds Timer-S1 (Liu et al., 2026c) on both metrics, despite Timer-S1 having 8.3B total parameters; Timer-S1 retains a lower GIFT-Eval MASE. PatchTST-FM-r1 has approximately 260M parameters, about 1.8 times as many as TABBY-PRETRAIN, and performs better on TIME.

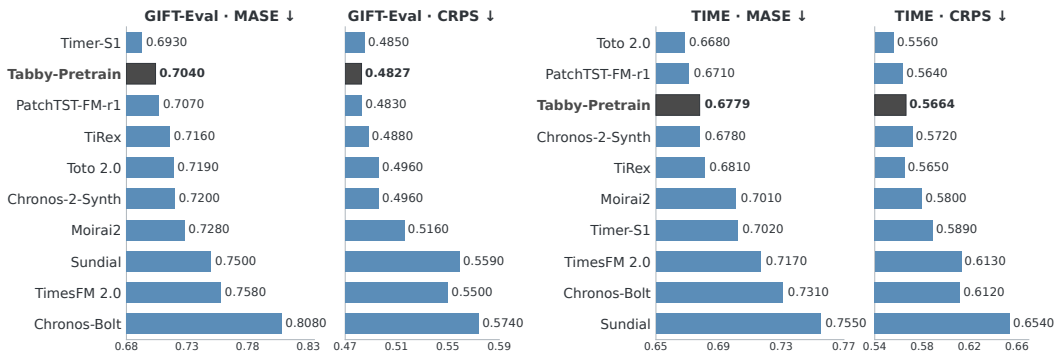


Figure 6: Zero-shot forecasting of the pretrained backbone on GIFT-Eval and TIME. The figure reports the selected baseline versions shown in the labels; lower values are better.

Toto 2.0 also leads on TIME, while TiRex has a slightly lower TIME CRPS. Overall, these results place our univariate backbone among the leading zero-shot forecasting models and show competitive performance against models with multivariate capabilities.

5.3 Transfer Beyond Forecasting

We next ask whether forecasting pretraining also yields representations that can be used for classification and anomaly detection. These experiments keep the encoder frozen but employ task-specific downstream procedures: a label-trained Random Forest for classification and a tuning-set-selected distance score for anomaly detection.

Table 3: Mean classification accuracy over all 128 UCR datasets.

	TabPFN	TabICL	TimesFM	MOMENT	TiRex	TiVIT-H	NuTime	MantisV1	Tabby-Pretrain
UCR	0.7806	0.7707	0.6765	0.7789	0.7457	0.7943	0.7732	0.7816	**0.7999**

Classification. We freeze the encoder and evaluate all 128 datasets in the UCR Archive (Dau et al., 2019). Each series is interpolated, resized to 1,024 time steps, right-aligned in the model window, and passed through the frozen Transformer. Padding-aware mean-pooled representations from all 20 layers are concatenated. A Random Forest is fitted using the official training labels of each dataset and evaluated once on its official test split. Thus, the encoder transfer is frozen, but the overall classification protocol is supervised rather than zero-shot.

With a label-trained Random Forest on the frozen encoder, Tabby-Pretrain reaches 0.7999 unweighted mean accuracy across all 128 UCR datasets (Table 3), above the listed baselines (Feofanov et al., 2026; Lin et al., 2024; Roschmann et al., 2025; Goswami et al., 2024; Qu et al., 2025; Hollmann et al., 2023) in this aggregate comparison. This supervised classification result requires neither classification-specific pretraining nor encoder adaptation.

Table 4: Mean VUS-PR over the TSB-AD-U evaluation split.

	Lag-Llama	Chronos	TimesFM	MOMENT (ZS)	MOMENT (FT)	xLSTMAD	Tabby-Pretrain
TSB-AD-U	0.27	0.27	0.30	0.38	0.39	0.40	**0.43**

Anomaly detection. We evaluate TSB-AD-U (Liu & Paparrizos, 2024) with VUS-PR (Paparrizos et al., 2022). Each series is tiled into non-overlapping windows, and final-block patch embeddings are pooled across windows. A Gaussian reference distribution is estimated separately from each sequence, and patches are scored by shrinkage Mahalanobis distance. Scores are broadcast to their time steps and min–max scaled. The backbone is frozen and no anomaly labels are used; the per-series statistics assume anomalies are a minority. The distance metric is selected on the 48-series tuning split, and all reported numbers use the held-out 350-series evaluation split. Details are given in Appendix F.

With a frozen encoder and a distance metric selected on the separate 48-series tuning split, Tabby-Pretrain attains mean VUS-PR 0.4282 on the held-out 350-series TSB-AD-U evaluation split. This exceeds the listed general-purpose foundation models (Rasul et al., 2024; Ansari et al., 2024;

Das et al., 2024; Goswami et al., 2024), xLSTMAD (Faber et al., 2025), and MOMENT in its fine-tuned form. Because adaptation and scoring differ across methods, this comparison supports the utility of the frozen representation.

6 Related Work

Time series forecasting has largely relied on task-specific models trained separately for individual datasets or application domains (Kim et al., 2025). Global forecasting models, such as DeepAR (Salinas et al., 2019), represented an important step toward sharing statistical information across related series, while architectures such as Triformer (Cirstea et al., 2022) and PatchTST (Nie et al., 2023) introduced patch-based representations for efficient forecasting. Nevertheless, these models were developed under dataset-specific training or downstream adaptation settings (Zeng et al., 2022). The emergence of time series foundation models has shifted this paradigm toward large scale pretraining followed by forecasting on datasets without task-specific parameter updates. Representative models, including TimeGPT (Garza et al., 2024), Lag-Llama (Rasul et al., 2024), TimesFM (Das et al., 2024), Chronos (Ansari et al., 2024), and Moirai (Woo et al., 2024), employ different architectures and prediction objectives but share the goal of learning transferable temporal patterns from large and heterogeneous time series corpora (Wang et al., 2025a;b).

More recent TSFMs have extended this paradigm along several dimensions. Models such as TiRex (Auer et al., 2025), Sundial (Liu et al., 2026b), and PatchTST-FM (Wen et al., 2026) support probabilistic forecasting through discrete sampling, parametric distributions, generative objectives, or direct quantile prediction. Moirai-2 (Liu et al., 2026a) and Chronos-2 (Ansari et al., 2025) extend zero-shot forecasting to multivariate series and covariate-informed settings, while TiRex-2 (Podest et al., 2026) further supports long-context and streaming inference. At the frontier of model scaling, Timer-S1 (Liu et al., 2026c) further pushes TSFMs to the billions parameter through a sparse mixture-of-experts architecture. Synthetic data have also become an important component of TSFM pretraining (Xie et al., 2026a; Oreshkin et al., 2026): ForecastPFN (Dooley et al., 2023) is trained entirely on synthetic series, whereas Toto 2.0 (Cohen et al., 2024; Khwaja et al., 2026) combine real-world data with synthetic processes designed to increase temporal and cross-variable diversity. To the best of our knowledge, PatchTST-FM (Wen et al., 2026) is the model most closely related to TABBY-PRETRAIN, as both adopt a channel independent patch Transformer, masked reconstruction, long input contexts, and dense quantile prediction. TABBY-PRETRAIN further introduces CauKerV2 as a structurally diverse synthetic data generator, a progressive convergence learning rate schedule, and deep supervision across intermediate transformer layers. In addition, we release the complete end-to-end pretraining codebase, covering data construction, synthetic data generation, distributed training, checkpoint continuation, and evaluation.

7 Discussion and Conclusion

This study has several limitations. TABBY-PRETRAIN is channel independent and does not explicitly model cross-variable dependencies or external covariates. Controlled comparisons are conducted primarily with TABBY-SMALL at one training budget, so they do not establish scaling laws or equal-FLOPs optimality. The restart experiment is a short local probe, and the DQS direction experiment concerns one checkpoint with trajectory regularization disabled. Finally, the CauKerV2 ablation supports complementarity within the present mixture but does not isolate individual generator families, SCM composition, or online sampling.

Within this scope, the evidence supports a practical answer to the motivating question. PCS organizes continuation into stages that end in annealed checkpoints, DQS supplies intermediate prediction signals, and CauKerV2 complements the real and kernel-generated training distribution. Fixed-budget Small experiments support the contribution of each component, while the full run shows stage-wise forecasting improvements and the final backbone supports forecasting, supervised classification from frozen representations, and anomaly detection. The open model and pipeline are intended to make these continuable pretraining choices reproducible and easier to extend.

AI USE STATEMENT

In this work, we used generative AI tools only to support writing and code organization. Specifically, we used them to edit the manuscript for readability and language quality, to suggest a structure for the paper, and to propose candidate titles. We also used them to organize the released codebase, e.g., refactoring, formatting, and documentation, without altering the implemented methods. We have not used generative AI tools to implement the model or its pretraining code, to clean, reformat, or modify datasets, to generate synthetic datasets, or to formulate mathematical claims. The remaining tasks requiring disclosure under the ICLR 2027 AI Policy, namely developing theoretical models or conceptual frameworks, writing proofs, proposing or refining hypotheses, designing research methodology or experiments, translation, qualitative data analysis, and interpreting results, were carried out without generative AI tools. All AI-assisted edits to the text and code were reviewed and verified by the authors, and every suggested title and structure was revised by the authors before adoption. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Bahaeddine Abdessalem, Shifeng Xie, Zehao Xiao, Youssef Attia El Hili, Ambroise Odonnat, Jianfeng Zhang, Lujia Pan, Keli Zhang, and Malik Tiomoko. Flowtsfm: Turning encoder depth into quantile transport, 2026. URL https://arxiv.org/abs/2609.13640.

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Gift-eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series, 2024. URL https://arxiv.org/abs/2403.07815.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting, 2025. URL https://arxiv.org/abs/2510.15821.

Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex: Zero-shot forecasting across long and short horizons with enhanced in-context learning, 2025. URL https://arxiv.org/abs/2505.23719.

Anthony Bao, Venkata Hasith Vattikuti, Jeffrey Lai, and William Gilpin. Universal redundancies in time series foundation models, 2026. URL https://arxiv.org/abs/2602.01605.

Manuel Burger, Fedor Sergeev, Malte Londschien, Daphné Chopard, Hugo Yèche, Eike Gerdes, Polina Leshetkina, Alexander Morgenroth, Zeynep Babür, Jasmina Bogojeska, Martin Faltys, Rita Kuznetsova, and Gunnar Rätsch. Towards foundation models for critical care time series, 2024. URL https://arxiv.org/abs/2411.16346.

Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting–full version, 2022. URL https://arxiv.org/abs/2204.13767.

Ben Cohen, Emaad Khwaja, Kan Wang, Charles Masson, Elise Ramé, Youssef Doubli, and Othmane Abou-Amal. Toto: Time series optimized transformer for observability, 2024. URL https://arxiv.org/abs/2407.07874.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, 2024. URL https://arxiv.org/abs/2310.10688.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive, 2019. URL https://arxiv.org/abs/1810.07758.

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha Naidu, and Colin White. Forecastpfn: Synthetically-trained zero-shot forecasting, 2023. URL https://arxiv.org/abs/2311.01933.

Kamil Faber, Marcin Pietron, Dominik Zurek, and Roberto Corizzo. xlstmad: A powerful xlstm-based method for anomaly detection. In *2025 IEEE International Conference on Data Mining (ICDM)*, pp. 247–256, 2025. doi: 10.1109/ICDM65498.2025.00032.

Vasilii Feofanov, Songkang Wen, Shifeng Xie, Simon Roschmann, Marius Alonso, Hongbo Guo, Romain Ilbert, Malik Tiomoko, Quentin Bouniot, Zeynep Akata, Lujia Pan, Jianfeng Zhang, and Ievgen Redko. Mantis: Lightweight foundation model for time series classification, 2026. URL https://arxiv.org/abs/2502.15637.

Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. Timegpt-1, 2024. URL https://arxiv.org/abs/2310.03589.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models, 2024. URL https://arxiv.org/abs/2402.03885.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second, 2023. URL https://arxiv.org/abs/2207.01848.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024. URL https://arxiv.org/abs/2404.06395.

Emaad Khwaja, Chris Lettieri, Gerald Woo, Eden Belouadah, Marc Cenac, Guillaume Jarry, Enguerrand Paquin, Xunyi Zhao, Viktoriya Zhukov, Othmane Abou-Amal, Chenghao Liu, Ameet Talwalkar, and David Asker. Toto 2.0: Time series forecasting enters the scaling era, 2026. URL https://arxiv.org/abs/2605.20119.

Jongseon Kim, Hyungjoon Kim, HyunGi Kim, Dongjun Lee, and Sungroh Yoon. A comprehensive survey of deep learning for time series forecasting: Architectural diversity and open challenges, 2025. URL https://arxiv.org/abs/2411.05793.

Siva Rama Krishna Kottapalli, Karthik Hubli, Sandeep Chandrashekhara, Garima Jain, Sunayana Hubli, Gayathri Botla, and Ramesh Doddaiah. Foundation models for time series: A survey, 2025. URL https://arxiv.org/abs/2504.04011.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets, 2014. URL https://arxiv.org/abs/1409.5185.

Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 6555–6565. ACM, August 2024. doi: 10.1145/3637528.3671451. URL http://dx.doi.org/10.1145/3637528.3671451.

Chenguo Lin, Xumeng Wen, Wei Cao, Congrui Huang, Jiang Bian, Stephen Lin, and Zhirong Wu. Nutime: Numerically multi-scaled embedding for large-scale time-series pretraining, 2024. URL https://arxiv.org/abs/2310.07402.

Chenghao Liu, Taha Aksu, Juncheng Liu, Xu Liu, Hanshu Yan, Quang Pham, Silvio Savarese, Doyen Sahoo, Caiming Xiong, and Junnan Li. Moirai 2.0: When less is more for time series forecasting, 2026a. URL https://arxiv.org/abs/2511.11698.

Qinghua Liu and John Paparrizos. The elephant in the room: Towards a reliable time-series anomaly detection benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 108231–108261. Curran Associates, Inc., 2024. doi: 10.52202/079017-3437. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c3f3c690b7a99fba16d0efd35cb83b2c-Paper-Datasets_and_Benchmarks_Track.pdf.

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models, 2026b. URL https://arxiv.org/abs/2502.00816.

Yong Liu, Xingjian Su, Shiyu Wang, Haoran Zhang, Haixuan Liu, Yuxuan Wang, Zhou Ye, Yang Xiang, Jianmin Wang, and Mingsheng Long. Timer-s1: A billion-scale time series foundation model with serial scaling, 2026c. URL https://arxiv.org/abs/2603.04791.

Zhen Liu, Boyuan Li, Hao Huang, Yanru Sun, Yucheng Wang, Min Wu, and Qianli Ma. From pre-training to post-training: A survey on time series foundation models. *TechRxiv*, 2026(0211), 2026d. doi: 10.36227/techrxiv.176978429.90235801/v2. URL https://www.techrxiv.org/doi/abs/10.36227/techrxiv.176978429.90235801/v2.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. URL https://arxiv.org/abs/1802.03426.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers, 2023. URL https://arxiv.org/abs/2211.14730.

Boris N. Oreshkin, Mayank Jauhari, Ravi Kiran Selvam, Malcolm Wolff, Wenhao Pan, Shankar Ramasubramanian, Kin G. Olivares, Tatiana Konstantinova, Andres Potapczynski, Mengfei Cao, Dmitry Efimov, Michael W. Mahoney, and Andrew G. Wilson. Zero-shot forecasting by simulation alone, 2026. URL https://arxiv.org/abs/2601.00970.

John Paparrizos, Paul Boniol, Themis Palpanas, Ruey S. Tsay, Aaron Elmore, and Michael J. Franklin. Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection. *Proc. VLDB Endow.*, 15(11):2774–2787, July 2022. ISSN 2150-8097. doi: 10.14778/3551793.3551830. URL https://doi.org/10.14778/3551793.3551830.

Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009. ISBN 052189560X.

Patrick Podest, Marco Pichler, Elias Bürger, Levente Zólyomi, Bernhard Voggenberger, Wilhelm Berghammer, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex-2: Generalizing tirex to multivariate data and streaming, 2026. URL https://arxiv.org/abs/2607.01204.

Zhongzheng Qiao, Sheng Pan, Anni Wang, Viktoriya Zhukova, Yong Liu, Xudong Jiang, Qingsong Wen, Mingsheng Long, Ming Jin, and Chenghao Liu. It's time: Towards the next generation of time series forecasting benchmarks. *arXiv preprint arXiv:2602.12147*, 2026.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabicl: A tabular foundation model for in-context learning on large data, 2025. URL https://arxiv.org/abs/2502.05564.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting, 2024. URL https://arxiv.org/abs/2310.08278.

Simon Roschmann, Quentin Bouniot, Vasilii Feofanov, Ievgen Redko, and Zeynep Akata. Time series representations for classification lie hidden in pretrained vision transformers, 2025. URL https://arxiv.org/abs/2506.08641.

David Salinas, Valentin Flunkert, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks, 2019. URL https://arxiv.org/abs/1704.04110.

Zezhi Shao, Yujie Li, Fei Wang, Chengqing Yu, Yisong Fu, Tangwen Qian, Bin Xu, Boyu Diao, Yongjun Xu, and Xueqi Cheng. Blast: Balanced sampling time series corpus for universal forecasting models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 2502–2513. ACM, August 2025. doi: 10.1145/3711896.3736860. URL http://dx.doi.org/10.1145/3711896.3736860.

Ege Onur Taga, M. Emrullah Ildiz, and Samet Oymak. Timepfn: Effective multivariate time series forecasting with synthetic data, 2025. URL https://arxiv.org/abs/2502.16294.

Yihang Wang, Yuying Qiu, Peng Chen, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. Lightgts: A lightweight general time series forecasting model, 2025a. URL https://arxiv.org/abs/2506.06005.

Yihang Wang, Yuying Qiu, Peng Chen, Kai Zhao, Yang Shu, Zhongwen Rao, Lujia Pan, Bin Yang, and Chenjuan Guo. Towards a general time series forecasting model with unified representation and adaptive transfer, 2025b. URL https://arxiv.org/abs/2405.17478.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Chen Wang, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark, 2026. URL https://arxiv.org/abs/2407.13278.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective, 2024. URL https://arxiv.org/abs/2410.05192.

Yunshi Wen, Wesley M. Gifford, Chandra Reddy, Lam M. Nguyen, Jayant Kalagnanam, and Anak Agung Julius. Revisiting the generic transformer: Deconstructing a strong baseline for time series foundation models, 2026. URL https://arxiv.org/abs/2602.06909.

Michał Wiliński, Mononito Goswami, Willa Potosnak, Nina Żukowska, and Artur Dubrawski. Exploring representations and interventions in time series foundation models, 2025. URL https://arxiv.org/abs/2409.12915.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers, 2024. URL https://arxiv.org/abs/2402.02592.

Shifeng Xie, Vasilii Feofanov, Ambroise Odonnat, Lei Zan, Marius Alonso, Jianfeng Zhang, Themis Palpanas, Lujia Pan, Keli Zhang, and Ievgen Redko. Cauker: Classification time series foundation models can be pretrained on synthetic data, 2026a. URL https://arxiv.org/abs/2508.02879.

Shifeng Xie, Ziwei Li, Themis Palpanas, Peilin Zhao, and Chenghao Liu. Time series foundation models improve LLM decisions: A case study in stock trading. In *1st ICLR Workshop on Time Series in the Age of Large Models*, 2026b. URL https://openreview.net/forum?id=XrGg8w03tJ.

Shifeng Xie, Ambroise Odonnat, Zehao Xiao, Lei Zan, Malik Tiomoko, Lujia Pan, Themis Palpanas, Boris N. Oreshkin, Chenghao Liu, and Keli Zhang. Post-training in time series foundation models: A unifying framework, 2026c. URL https://arxiv.org/abs/2607.20002.

Javier Yanes-Pulido and Filipe Rodrigues. Time series foundation models as strong baselines in transportation forecasting: A large-scale benchmark analysis, 2026. URL https://arxiv.org/abs/2602.24238.

Tahar Zanouda, Meysam Masoudi, Fitsum Gaim Gebre, and Mischa Dohler. Telecom foundation models: Applications, challenges, and future trends, 2024. URL https://arxiv.org/abs/2408.03964.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting?, 2022. URL https://arxiv.org/abs/2205.13504.

A Model Architecture Details

A.1 Input window and mask-aware normalization

Let

$$n_o = \sum_{t=1}^{L_{\max}} o_t, \qquad \bar{n}_o = \max(n_o, 1),$$

and compute

$$\mu = \frac{1}{\bar{n}_o} \sum_{t=1}^{L_{\max}} o_t x_t, \qquad v = \frac{1}{\bar{n}_o} \sum_{t=1}^{L_{\max}} o_t (x_t - \mu)^2.$$

The scale used by the released training and evaluation implementation is

$$\bar{s} = \begin{cases} \sqrt{v + \epsilon}, & n_o \geq 2, \\ 1, & n_o < 2, \end{cases} \qquad \epsilon = 10^{-5}.$$

The normalized input and inverse transformation are

$$z_t = \operatorname{asinh}\left(\frac{x_t - \mu}{\bar{s}}\right), \qquad \widetilde{z}_t = o_t z_t,$$

$$\hat{x}_{t,\tau} = \mu + \bar{s} \sinh(\hat{z}_{t,\tau}).$$

A.2 Patch encoder

The window is split into $N = L/P$ non-overlapping patches of length P. For patch n, we concatenate its normalized values and visibility indicators:

$$\mathbf{u}_n = [\widetilde{\mathbf{z}}_n; \mathbf{o}_n] \in \mathbb{R}^{2P}. \tag{1}$$

Each patch is mapped to the model dimension by a residual two-layer projection,

$$\phi_{\text{in}}(\mathbf{u}) = \mathbf{W}_2 \, \sigma(\mathbf{W}_1 \mathbf{u} + \mathbf{b}_1) + \mathbf{b}_2 + \mathbf{W}_r \mathbf{u} + \mathbf{b}_r, \tag{2}$$

where σ is the sigmoid function. The initial representation is

$$\mathbf{h}_n^{(0)} = \phi_{\text{in}}(\mathbf{u}_n) + \mathbf{e}_n, \tag{3}$$

where $\mathbf{e}_n \in \mathbb{R}^d$ is a learned absolute positional embedding. Padded patches are excluded as attention keys, while attention among all non-padded patches is bidirectional and non-causal.

A.3 Transformer encoder

The encoder contains M pre-normalization Transformer blocks. For layer $\ell = 1, \ldots, M$,

$$\mathbf{a}^{(\ell)} = \mathbf{h}^{(\ell-1)} + \text{MHA}\Big(\text{LN}(\mathbf{h}^{(\ell-1)})\Big), \tag{4}$$

$$\mathbf{h}^{(\ell)} = \mathbf{a}^{(\ell)} + \text{Dropout}\Big(\text{FFN}\Big(\text{LN}(\mathbf{a}^{(\ell)})\Big)\Big). \tag{5}$$

Multi-head self-attention uses $n_h = d/d_h$ heads. The feed-forward network has hidden width $4d$ and a GELU activation. No causal mask is used; information leakage is prevented by zeroing the withheld values and providing only their visibility indicators to the encoder.

A.4 Monotone quantile decoder

Let K be the number of predicted quantiles. A residual output projection maps each final patch representation to $P(K+1)$ raw values. After reshaping, let $r_{n,j,0}, \ldots, r_{n,j,K}$ denote the raw outputs for offset j within patch n. The normalized quantile predictions are

$$\widehat{z}_{n,j,k} = r_{n,j,0} + \sum_{i=1}^{k} \frac{\text{softplus}(r_{n,j,i})}{K}, \qquad k = 1, \ldots, K. \tag{6}$$

Table 5: Architecture configuration of TABBY-PRETRAIN.

Hyperparameter	Value
Complete model window L	8192
Patch length P	16
Number of patch tokens $N = L/P$	512
Transformer layers M	20
Model dimension d	768
Attention heads n_h	12
Head dimension d_h	64
Feed-forward width	3072
Dropout	0.1
Quantile levels K	99
Pretraining mask ratio	0.4
Contiguous-mask setting B_{cpm}	8 patches
Normalization stabilizer ϵ	10^{-5}

Because every increment is positive, $\widehat{z}_{n,j,k+1} \geq \widehat{z}_{n,j,k}$, so quantile crossing cannot occur. With $K = 99$, the modeled levels are

$$\tau_k = \frac{k}{K+1} = \frac{k}{100}, \qquad k = 1, \ldots, 99, \tag{7}$$

covering 0.01 through 0.99. These outputs represent pointwise conditional marginal quantiles; the model does not define a joint distribution over the forecast horizon.

A.5 FORECAST-WINDOW CONSTRUCTION AND MULTIVARIATE INPUTS

For a requested forecast horizon H, the inference wrapper reserves

$$F = \max(H,\ P \max(B_{\mathrm{cpm}}, 2)) = \max(H, 128) \tag{8}$$

masked positions, where $P = 16$ and $B_{\mathrm{cpm}} = 8$. It retains at most $L - F$ recent observations, left-pads shorter inputs, appends the F masked positions, and returns the first H forecasts. Consequently, $L = 8192$ is the total model window rather than the number of historical observations always available to the encoder.

For a multivariate input $\mathbf{X} \in \mathbb{R}^{B \times L \times D_v}$, the implementation reshapes the input to $(BD_v) \times L$, applies the same univariate backbone to every channel, and then restores the channel dimension. Normalization statistics are computed separately for each channel. Hence, all channels share model parameters, but there is no cross-channel attention in the current architecture.

The inference interface optionally provides a single-value summary of the predicted quantiles using symmetric triangular weights:

$$\widehat{y}_t^{\mathrm{tri}} = \sum_{k=1}^{K} w_k \widehat{q}_{t,\tau_k}, \qquad w_k = \frac{0.5 - |0.5 - \tau_k|}{\sum_{i=1}^{K} (0.5 - |0.5 - \tau_i|)}. \tag{9}$$

This aggregation assigns larger weights to quantiles near the center of the predictive distribution. It is provided as an optional point summary and should not be interpreted as the predictive mean. In all forecasting results reported in this work, including MASE, we instead use the median forecast,

$$\widehat{y}_t^{\mathrm{median}} = \widehat{q}_{t,0.5}. \tag{10}$$

A.6 CONFIGURATION AND PARAMETER COUNT

The exact parameter decomposition is reported in Table 6. The input projection receives $2P = 32$ features per patch, and the output projection produces $P(K+1) = 1600$ raw values per patch.

The state dictionary contains 253 parameter tensors. Storing the parameters alone requires approximately 556.348 MiB in FP32 or 278.174 MiB in BF16; these figures exclude gradients, optimizer states, and activations.

Table 6: Parameter decomposition of TABBY-PRETRAIN.

Component	Parameters	Share
Input residual projection	641,280	0.44%
Learned positional embedding	393,216	0.27%
20 Transformer blocks	141,757,440	97.20%
Output residual projection	3,051,392	2.09%
Total	145,843,328	100.00%

B GIFT-EVAL-PRETRAIN+: REAL-WORLD DATA CORPORA

Corpus scope. GIFT-Eval-Pretrain+ is the curated real-world component used in our pretraining corpora. It starts from the stored GIFT-Eval-Pretrain collection (Aksu et al., 2024), restores a small set of eligible real-world series removed during earlier filtering, and adds complementary datasets selected from the TiRex-2 training corpora (Podest et al., 2026). The final on-disk corpus contains 150 source-dataset entries organized into 134 training sampling units. The difference arises because several small, semantically related datasets are stored in a shared directory and distinguished by a `source_dataset` field. These consolidated directories are sampled as single training units under the balanced sampling scheme described in the main text.

Counting convention. For dataset d, let N_d denote the number of stored Hugging Face rows (samples). If sample n contains C_{dn} channels of length T_{dn}, we report

$$C_d = \sum_{n=1}^{N_d} C_{dn}, \qquad P_d = \sum_{n=1}^{N_d} C_{dn} T_{dn}, \qquad \overline{T}_d = \frac{P_d}{C_d}.$$

Here, C_d is the number of univariate channel sequences, P_d is the number of scalar observations, and $\overline{T}_d$ is the channel-weighted mean sequence length. This distinction is necessary because one stored sample may contain multiple channels. In Table 9, U denotes entries with $C_d = N_d$, while M denotes entries with $C_d > N_d$. The U/M flag therefore describes the stored target representation rather than the forecasting protocol used downstream.

Overall characteristics. The corpus contains 3,792,438 samples, 28,872,274 channel sequences, and 229,802,174,843 scalar observations. The global channel-weighted mean length is 7,959.27 observations. Dataset-level mean lengths range from 48.95 to 538,725.31 observations, covering both short forecasting collections and very long sensor or power-generation records. Seventy-three stored dataset entries contain multichannel samples; together, they contribute 203,407,137,243 observations (88.5% of the corpus). The collection spans energy, climate and environment, transportation, web and cloud operations, sales, economics and finance, healthcare, tourism, and natural phenomena, with sampling intervals ranging from seconds to quarterly observations. The scan found no malformed targets.

All counts below describe the post-curation files actually used by our data loader; they need not match the raw upstream releases because filtering, restoration, sharding, and channel expansion are applied before pretraining.

B.1 CONSOLIDATION INTO TRAINING SAMPLING UNITS

Uniform sampling over source datasets can severely oversample very small collections. We therefore consolidate selected small datasets before constructing the balanced sampler. The consolidation is performed only at the sampling-unit level: the original dataset identity is retained in the `source_dataset` field, and all samples remain individually recoverable. Table 7 reports every composite unit found in the final training manifest.

The first three units group datasets from the same or closely related application areas. The recovered-real unit instead groups datasets by curation provenance: these datasets were restored after an earlier filtering stage and are not assumed to belong to a single application domain.

Table 7: Source datasets consolidated into shared training sampling units. The row count is the number of stored Hugging Face samples in the composite unit.

Training sampling unit	Constituent source datasets	Rows
energy_covariates_small	cockatoo, covid19_energy, gfc14_load, pdb, spain, and spanish_energy_and_weather	129
macro_finance_small	bitcoin_with_missing, exchange_rate, and monash_fred_md	321
public_health_small	cdc_fluview_ilinet, cdc_fluview_who_nrevss, and covid_mobility	1,304
recovered_useful_real	HZMETRO, SHMETRO, borealis, elecdemand, elf, smart, solar_power, and wind_power	4,096

Table 8: Composition of GIFT-Eval-Pretrain+ by collection source. Mean length is computed as total observations divided by total channel sequences.

Code	Collection component	Entries	Samples	Channels	Observations
G	Stored GIFT-Eval subset	120	3,782,761	28,855,468	228,829,433,709
T	TiRex-2 supplement	21	5,474	12,603	943,028,306
R	Restored curated subset	9	4,203	4,203	29,712,828
Total	GIFT-Eval-Pretrain+	150	3,792,438	28,872,274	229,802,174,843

Frequency notation. *s*, *min*, *h*, *d*, *wk*, *mo*, and *qtr* denote seconds, minutes, hours, days, weeks, months, and quarters, respectively. Frequencies record the nominal sampling interval of the stored source dataset; they are not resampling factors applied by the model.

Table 9: Complete dataset-level inventory of GIFT-Eval-Pretrain+. Source codes follow Table 8. Samples are stored rows; channels are univariate sequences after expanding the target dimension; mean length is P_d/C_d; and observations count every scalar value across all channels.

Dataset	Src.	Domain	Freq.	Type	Samples	Channels	Mean length	Observations
BEIJING_SUBWAY_30MIN	T	Transport	30 min	M	276	552	1,572.00	867,744
PEMS04	T	Transport	5 min	M	307	921	16,992.00	15,649,632
PEMS08	T	Transport	5 min	M	170	510	17,856.00	9,106,560
beijing_air_quality	T	Environment	1 h	M	12	132	35,064.00	4,628,448
china_air_quality	T	Environment	1 h	M	437	2,622	13,133.26	34,435,404
cockatoo	T	Energy	1 h	U	6	6	17,544.00	105,264
covid19_energy	T	Energy	1 h	U	14	14	31,912.00	446,768
gfc14_load	T	Energy	1 h	U	1	1	17,520.00	17,520
pdb	T	Energy	1 h	U	2	2	17,520.00	35,040
spain	T	Energy	1 h	U	4	4	35,064.00	140,256
spanish_energy_and_weather	T	Energy	1 h	U	102	102	35,064.00	3,576,528
bitcoin_with_missing	R	Econ./finance	1 d	U	107	107	4,551.00	486,957
exchange_rate	T	Econ./finance	1 d	U	107	107	7,588.00	811,916
monash_fred_md	T	Econ./finance	1 mo	U	107	107	728.00	77,896
project_tycho	T	Healthcare	1 wk	U	1,258	1,258	1,095.16	1,377,707
cdc_fluview_ilinet	T	Healthcare	1 wk	U	646	646	887.00	573,000
cdc_fluview_who_nrevss	T	Healthcare	1 wk	U	296	296	564.32	167,040
covid_mobility	T	Transport	1 d	U	362	362	410.50	148,602
HZMETRO	R	Transport	15 min	U	512	512	2,377.00	1,217,024
SHMETRO	R	Transport	15 min	U	512	512	8,192.00	4,194,304
borealis	R	Energy	1 h	U	512	512	5,552.78	2,843,023
elecdemand	R	Energy	30 min	U	512	512	8,192.00	4,194,304
elf	R	Energy	1 h	U	512	512	8,192.00	4,194,304
smart	R	Energy	1 h	U	512	512	8,192.00	4,194,304
solar_power	R	Energy	4 s	U	512	512	8,192.00	4,194,304
wind_power	R	Energy	4 s	U	512	512	8,192.00	4,194,304
residential_load_power	T	Energy	1 min	M	271	813	538,725.31	437,983,677
residential_pv_power	T	Energy	1 min	M	233	699	537,935.41	376,016,850
subseasonal	T	Climate	1 d	M	862	3,448	16,470.00	56,788,560
sunspot_with_missing	T	Nature	1 d	U	1	1	73,894.00	73,894
mexico_city_bikes	G	Transport	1 h	U	494	494	78,313.77	38,687,004
ushcn_daily	G	Climate	1 d	U	1,218	1,218	38,653.62	47,080,115
weatherbench_daily	G	Climate	1 d	U	225,280	225,280	14,609.98	3,291,336,704
weatherbench_weekly	G	Climate	1 wk	U	225,280	225,280	2,087.00	470,159,360
LOS_LOOP	G	Transport	5 min	U	207	207	34,272.00	7,094,304
PEMS03	G	Transport	5 min	U	358	358	26,208.00	9,382,464
PEMS07	G	Transport	5 min	U	883	883	28,224.00	24,921,792
PEMS_BAY	G	Transport	5 min	U	325	325	52,128.00	16,941,600
Q-TRAFFIC	G	Transport	15 min	U	45,148	45,148	5,856.00	264,386,688
alibaba_cluster_trace_2018	G	Cloud operations	5 min	M	58,409	116,818	1,629.76	190,385,060

Continued on next page

Table 9 – continued from previous page

Dataset	Src.	Domain	Freq.	Type	Samples	Channels	Mean length	Observations
australian_electricity_demand	G	Energy	30 min	U	5	5	230,716.80	1,153,584
azure_vm_traces_2017	G	Cloud operations	5 min	U	159,472	159,472	5,552.84	885,522,908
bdg-2_bear	G	Energy	1 h	U	91	91	16,289.14	1,482,312
bdg-2_fox	G	Energy	1 h	U	135	135	17,219.02	2,324,568
bdg-2_panther	G	Energy	1 h	U	105	105	8,760.00	919,800
bdg-2_rat	G	Energy	1 h	U	280	280	16,886.74	4,728,288
borg_cluster_data_2011	G	Cloud operations	5 min	M	143,386	286,772	3,748.99	1,075,105,708
buildings_900k	G	Energy	1 h	U	1,795,256	1,795,256	8,761.00	15,728,237,816
bull	G	Energy	1 h	U	41	41	17,544.00	719,304
cmip6_1850	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1855	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1860	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1865	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1870	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1875	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1880	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1885	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1890	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1895	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1900	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1905	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1910	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1915	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1920	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1925	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1930	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1935	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1940	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1945	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1950	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1955	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1960	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1965	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1970	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1975	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1980	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1985	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1990	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_1995	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_2000	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_2005	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
cmip6_2010	G	Climate	6 h	M	8,192	434,176	7,300.00	3,169,484,800
era5_1989	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040

Continued on next page

Table 9 – continued from previous page

Dataset	Src.	Domain	Freq.	Type	Samples	Channels	Mean length	Observations
era5_1990	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_1991	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_1992	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_1993	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_1994	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_1995	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_1996	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_1997	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_1998	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_1999	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2000	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2001	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2002	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2003	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2004	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2005	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2006	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2007	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2008	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2009	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2010	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2011	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2012	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2013	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2014	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2015	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2016	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2017	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
era5_2018	G	Climate	1 h	M	8,192	368,640	8,736.00	3,220,439,040
extended_web_traffic_with_missing	G	Web	1 d	U	145,063	145,063	2,557.00	370,926,091
favorita_sales	G	Sales/retail	1 d	U	111,840	111,840	1,244.45	139,179,538
favorita_transactions	G	Sales/retail	1 d	U	54	54	1,563.11	84,408
gfc12_load	G	Energy	1 h	U	20	20	39,414.00	788,280
gfc17_load	G	Energy	1 h	U	8	8	17,544.00	140,352
hog	G	Energy	1 h	U	24	24	17,544.00	421,056
ideal	G	Energy	1 h	U	217	217	5,784.58	1,255,253
kaggle_web_traffic_weekly	G	Web	1 wk	U	145,063	145,063	114.00	16,537,182
kdd2022	G	Energy	10 min	U	134	134	35,279.99	4,727,519
largest_2017	G	Transport	5 min	U	8,196	8,196	105,120.00	861,563,520
largest_2018	G	Transport	5 min	U	8,428	8,428	105,120.00	885,951,360
largest_2019	G	Transport	5 min	U	8,600	8,600	105,120.00	904,032,000
largest_2020	G	Transport	5 min	U	8,561	8,561	105,408.00	902,397,888
largest_2021	G	Transport	5 min	U	8,548	8,548	105,120.00	898,565,760

Continued on next page

Table 9 – continued from previous page

Dataset	Src.	Domain	Freq.	Type	Samples	Channels	Mean length	Observations
lcl	G	Energy	1 h	U	713	713	13,385.07	9,543,553
london_smart_meters_with_missing	G	Energy	30 min	U	5,520	5,520	30,115.74	166,238,880
m1_monthly	G	Econ./finance	1 mo	U	617	617	72.76	44,892
m5	G	Sales/retail	1 d	U	30,490	30,490	1,913.00	58,327,370
monash_m3_monthly	G	Econ./finance	1 mo	U	1,428	1,428	99.34	141,858
monash_m3_other	G	Econ./finance	1 qtr	U	174	174	68.58	11,933
monash_m3_quarterly	G	Econ./finance	1 qtr	U	756	756	48.95	37,004
nn5_daily_with_missing	G	Econ./finance	1 d	U	111	111	735.00	81,585
nn5_weekly	G	Econ./finance	1 wk	U	111	111	105.00	11,655
oikolab_weather	G	Climate	1 h	U	8	8	100,057.00	800,456
pedestrian_counts	G	Transport	1 h	U	66	66	47,435.79	3,130,762
rideshare_with_missing	G	Transport	1 h	U	2,304	2,304	373.00	859,392
subseasonal_precip	G	Climate	1 d	U	862	862	11,323.00	9,760,426
taxi_30min	G	Transport	30 min	U	67,984	67,984	809.00	54,999,056
tourism_monthly	G	Tourism	1 mo	U	366	366	274.58	100,496
tourism_quarterly	G	Tourism	1 qtr	U	427	427	91.63	39,128
traffic_hourly	G	Transport	1 h	U	862	862	17,376.00	14,978,112
traffic_weekly	G	Transport	1 wk	U	862	862	96.00	82,752
uber_tlc_daily	G	Transport	1 d	U	262	262	179.72	47,087
uber_tlc_hourly	G	Transport	1 h	U	262	262	4,310.85	1,129,444
vehicle_trips_with_missing	G	Transport	1 d	U	329	329	98.82	32,512
weather	G	Climate	1 d	U	3,010	3,010	14,266.35	42,941,700
wiki-rolling_nips	G	Web	1 d	U	47,675	47,675	852.00	40,619,100
wind_farms_with_missing	G	Energy	1 min	U	337	337	510,876.47	172,165,370

C CONFIGURATION OF TABBY-SMALL

To reduce the computational cost of controlled ablation studies, we construct TABBY-SMALL by reducing the depth and hidden dimension of the full model while retaining its long-context input representation, patch-based encoder, probabilistic quantile decoder, masking objective, and pretraining data mixture. The resulting model contains exactly 40,291,456 parameters (40.291M), measured directly from the checkpoint at step 110,000. The checkpoint contains 157 parameter tensors with a raw weight size of 153.700 MiB.

The model is trained for 110,000 optimization steps on eight GPUs. With a per-device batch size of 64 and no gradient accumulation, its global batch size is

$$B_{\text{global}} = 8 \times 64 \times 1 = 512.$$

The total number of sampled training windows is therefore

$$N_{\text{windows}} = 110{,}000 \times 512 = 56{,}320{,}000.$$

This value represents the number of sample exposures during optimization rather than the number of unique time series, because the stored corpora are repeatedly sampled and CauKerV2 generates synthetic samples online. The complete configuration is reported in Table 10.

Table 10: Complete architecture, data, masking, supervision, and optimization configuration of TABBY-SMALL.

Configuration	**Value**
Model architecture	
Context length	8,192
Patch size	16
Maximum number of patch tokens	512
Number of Transformer layers	12
Hidden dimension	512
Attention-head dimension	32
Number of attention heads	16
MLP expansion ratio	4
MLP hidden dimension	2,048
Dropout probability	0.10
Number of predicted quantiles	99
Number of parameter tensors	157
Number of parameters	40,291,456
Number of parameters in millions	40.291M
Raw checkpoint weight size	153.700 MiB
Pretraining data mixture	
BLAST sampling ratio	0.35
KernelSynth sampling ratio	0.30
GIFT-Eval-Pretrain+ sampling ratio	0.25
CauKerV2 sampling ratio	0.10
Mixup sampling ratio	0
Minimum general real-series length	64
BLAST sequence-length range	$[96, 4096]$
KernelSynth sequence-length range	$[96, 8192]$
GIFT-Eval-Pretrain+ sequence-length range	$[2048, 8192]$
CauKerV2 sequence-length range	$[96, 2048]$
Sequence-length sampling strategy	Uniform
Length-weighting coefficient	0.5
GIFT-Eval-Pretrain+ balancing	Dataset/domain balanced
KernelSynth shard balancing	Enabled
CauKerV2 configuration	
Generation mode	Online
Number of generated features	12
Number of SCM nodes	12

Continued on the next page

Table 10 continued

Configuration	Value
Maximum number of parents per node	2
Maximum generated sequence length	2,048
Data-loading configuration	
Number of BLAST workers	3
Number of KernelSynth workers	3
Number of GIFT-Eval-Pretrain+ workers	3
Number of mixup workers	0
Prefetch factor	4
Masking configuration	
Overall mask ratio	0.40
Number of CPM blocks	8
Minimum terminal-mask length	0 patches
Maximum terminal-mask length	3 patches
Deep-supervision configuration	
Deep supervision	Enabled
Number of supervised exits	5
Decoded exit layers	$\{0, 3, 6, 9, 12\}$
Deep-supervision weight λ_{DS}	0.5
Trajectory-regularization weight λ_{TRA}	0.1
Depth-weighting exponent γ	1.0
Optimization configuration	
Total optimization steps	110,000
Number of GPUs	8
Per-device batch size	64
Gradient-accumulation steps	1
Global batch size	512
Total sampled training windows	56.32M
Training precision	bfloat16
Optimizer	AdamW
First-moment coefficient β_1	0.9
Second-moment coefficient β_2	0.95
Weight decay	0.1
Maximum gradient norm	1.0
Learning-rate schedule	
Scheduler	Multi-stage WSD
Number of stable–decay cycles	2
Warm-up steps	10,000
Decay steps	10,000
Peak learning rate	2×10^{-4}
Minimum learning rate	1×10^{-5}
Checkpointing and logging	
Checkpoint interval	5,000 steps
Final analyzed checkpoint	Step 110,000
Logging interval	100 steps
Training log format	JSON Lines

All controlled comparisons in the analysis use this configuration as their reference setting. For each ablation, only the component under investigation is modified; the model size, pretraining data mixture, global batch size, optimization horizon, and all remaining hyperparameters are held fixed.

C.1 Terminal-Mask Length Analysis

We examine the effect of longer extrapolation targets during pretraining by increasing the reported effective maximum terminal-mask length from 2 to 32 patches. Both variants use TABBY-SMALL and the same 110K-step training budget, with all remaining settings held fixed.

Table 11: Terminal-mask ablation on GIFT-Eval. Lower is better.

Maximum patches	MASE	CRPS
32	0.7473	0.5116
2	**0.7368**	**0.5038**

The longer terminal targets raise MASE by 1.43% and CRPS by 1.55% at the same budget. This result supports the short-terminal-mask setting used in the present recipe; it does not by itself establish that longer targets cause optimization instability.

D Synthetic Data Generation and Hyperparameters

Scope. The synthetic component of the final training sampler consists of two distinct streams: pre-generated KernelSynth series and CauKerV2 series generated online. Their sampling probabilities are 0.30 and 0.10, respectively; the remaining 0.60 is assigned to the two real-world streams. These probabilities specify how training examples are drawn and should not be interpreted as the relative sizes of stored datasets. KernelSynth is finite and stored in Arrow files, whereas CauKerV2 is an infinite iterable stream whose samples are generated independently by each data-loader worker.

The values below are taken directly from the supplied implementation. We distinguish fixed generator distributions from constructor defaults. The latter can be overridden by the training command and therefore do not, by themselves, establish the values used by the final checkpoint.

D.1 KernelSynth

Coarse-grid generation. For a requested target length T, our KernelSynth construction generates a Gaussian-process realization on a grid that is four times coarser and then upsamples it to the target grid. Let $r = 4$ and $T_{\mathrm{c}} \simeq T/r$. The stored sequence is

$$\widetilde{\mathbf{x}}_{1:T} = \mathcal{I}_r\big(\mathbf{x}^{\mathrm{GP}}_{1:T_{\mathrm{c}}}\big)\,,$$

where $\mathcal{I}_r$ denotes the interpolation operator. Reducing the covariance grid from T to approximately $T/4$ lowers a quadratic covariance-construction term by

$$\frac{T_{\mathrm{c}}^2}{T^2} \simeq \frac{1}{16}.$$

Independent output shards are generated by separate CPU tasks and stored in the GluonTS Arrow format.

Arrow sampling. At training time, all `.arrow` files are discovered recursively. With file balancing enabled, a worker shuffles the complete file list and draws one valid record from each file before reshuffling. Within a file, records are selected uniformly by random access when available. A failed or malformed record is rejected, with at most 32 attempts for the selected file. The extracted sequence is passed to the same formatting routine used for real data, which performs subwindow sampling and left padding to the model context.

Table 12: KernelSynth reader defaults. These are implementation defaults; final launch-time overrides are not shown.

Parameter	Default	Meaning
`context_length`	2,048×4	Padded model input length
`min_sample_length`	96	Minimum sampled training length
`max_sample_length`	8,192	Maximum sampled training length

Table 13: Primitive generator families used by CauKerV2. Representative domains are illustrative rather than exhaustive.

Generator family	Temporal characteristics	Representative domains
Compositional-kernel Gaussian process	Smoothness and local correlations	Weather, energy, environmental sensing
Trend–seasonality decomposition	Additive or multiplicative trends and seasonality	Retail, traffic, electricity demand
ARIMA/SARIMA state-space process	Linear dependence, integration, seasonal autoregression	Economics, demand, climate
Regime-switching Ornstein–Uhlenbeck SDE	Mean reversion, stochasticity, regime transitions	Finance, physical systems
Piecewise level and change-point process	Abrupt or smooth shifts and piecewise stationarity	Industrial monitoring, operations
Spike and event process	Spikes, bursts, shocks, plateaus, recovery	Web traffic, energy, healthcare
Analytic waveform mixture	Sawtooth, square, and triangular periodic signals	Sensors, machinery, electronics
Fractional Brownian motion/noise	Long-range dependence and scale-dependent stochasticity	Network traffic, geophysics, finance
GARCH-family volatility process	Volatility clustering, heavy tails, asymmetric shocks	Financial and energy markets
Chaotic and audio-inspired process	Lorenz, Mackey–Glass, NARMA, multi-scale oscillations	Physical, audio, biomedical signals

D.2 CauKerV2

Online generation pipeline. For each multivariate draw, CauKerV2 first samples a sequence length L and constructs K primitive root series,

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K] \in \mathbb{R}^{L \times K}.$$

The primitive family assigned to each column is selected by a weighted schedule. A random directed acyclic graph then transforms these roots into P observed series,

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_P] \in \mathbb{R}^{L \times P}.$$

The current pretraining objective is channel independent. The P output channels are therefore shuffled in a local buffer and yielded separately as P univariate training examples. Each sequence is placed at the end of a zero-initialized context window, with left padding marked by the padding mask.

Length sampling. The default length sampler is log-uniform. For lower and upper bounds $L_{\min}$ and $L_{\max}$,

$$\log L \sim \mathcal{U}(\log L_{\min}, \log L_{\max}), \qquad L = \mathrm{clip}\big(\mathrm{round}(e^{\log L}), L_{\min}, L_{\max}\big)\,.$$

Uniform integer sampling is also implemented as an alternative. Invalid numerical draws are rejected and regenerated without terminating the iterable stream.

Table 14: Constructor defaults of `OnlineCauKerIterableDataset`. Values passed by the final launch command override this table.

Parameter	Default	Meaning
`context_length`	2,048	Padded model input length
$L_{\min}$: `min_length`	96	Minimum generated length
$L_{\max}$: `max_length`	2,048	Maximum generated length
P: `num_features`	8	Observed SCM channels
K: `num_nodes`	8	Primitive root series (latent dimension)
M: `max_parents`	6	Maximum parents of an observed node
`length_sampling`	log-uniform	Distribution over L
Normalization clip	8	Robust-normalization clipping threshold

Numerical ϵ	10^{-6}	Minimum admissible standard deviation

D.2.1 PRIMITIVE GENERATOR PORTFOLIO

The default portfolio contains ten generator families. Let w_f be the weight assigned to family f. For a draw with K roots, the implementation first allocates $\lfloor Kw_f \rfloor$ roots, applies a minimum-one heuristic when $K \geq 10$, corrects any resulting over-allocation, distributes the remaining roots according to the largest fractional remainders, and finally shuffles the schedule. The over-allocation correction may remove a previously assigned minimum-one slot. Under the default weights in Table 15, all ten families are simultaneously represented only from $K = 12$ onward; $K = 10$ or 11 can still omit low-weight families. The constructor default $K = 8$ therefore does not provide full per-draw coverage.

Table 15: Default primitive-family weights used by CauKerV2. The weights sum to one.

Generator family	Weight
Compositional-kernel Gaussian process	0.10
Trend–seasonality decomposition	0.10
Regime-switching Ornstein–Uhlenbeck SDE	0.10
ARIMA/SARIMA state-space process	0.10
Piecewise level and change-point process	0.10
Spike and event process	0.30
Analytic waveform mixture	0.05
Fractional Brownian motion/noise	0.05
GARCH-family volatility process	0.05
Chaotic and audio-inspired process	0.05

Table 16: Code-level hyperparameter distributions for the ten CauKerV2 primitive generators. $\mathcal{U}(a, b)$ denotes a continuous uniform distribution and $\mathrm{LogU}(a, b)$ a log-uniform distribution.

Family	Implemented distributions and choices
Compositional-kernel GP	Select $m \sim \mathrm{UnifInt}\{1, \ldots, 7\}$ kernels with replacement from a 33-entry bank. Periodic kernels use normalized periods $\{24, 48, 96, 168, 336, 672, 7, 14, 30, 60, 365, 730, 4, 26, 52, 4, 6, 12, 4, 40, 10\}/L$; dot-product kernels use $\sigma_0 \in \{0, 1, 10\}$; RBF length scales are $\{0.1, 1, 10\}$; rational-quadratic α values are $\{0.1, 1, 10\}$; white-noise levels are $\{0.1, 1\}$; and one constant kernel is included. Successive kernels are combined by addition or multiplication with probability 0.5 each. Two mean functions are sampled with replacement from zero, linear, exponential, and sparse-anomaly means and are also added or multiplied with probability 0.5. Linear slopes/intercepts follow $\mathcal{U}(-1, 1)$; exponential amplitude/rate follow $\mathcal{U}(0.5, 1.5)$; the anomaly mean contains 1–5 impulses with amplitudes $\mathcal{U}(-5, 5)$. GP covariance jitter is 10^{-6}.
Trend–seasonality	Trend type probabilities for none, linear, quadratic, exponential, damped, and piecewise trends are $(0.08, 0.27, 0.20, 0.15, 0.15, 0.15)$. Linear slopes are $\mathcal{U}(-3, 3)$; quadratic coefficients use $\mathcal{U}(-4, 4)$ and $\mathcal{U}(-2, 2)$; exponential rate is $\mathcal{U}(-3, 3)$ with amplitude $\mathcal{U}(0.3, 2)$; damped rate is $\mathcal{U}(1, 8)$. The number of seasonal components is uniform in $\{0, 1, 2, 3\}$, with candidate periods $\{4, 6, 7, 12, 24, 48, 52, 96, 168, 336\}$, amplitude $\mathcal{U}(0.2, 2)$, and phase $\mathcal{U}(0, 2\pi)$. Seasonal waveform probabilities for sine, triangle, step, and impulse are $(0.55, 0.20, 0.15, 0.10)$. Irregular-component probabilities for white, colored, random-walk, and fBm-like noise are $(0.35, 0.35, 0.15, 0.15)$. Additive, multiplicative, and mixed composition probabilities are $(0.55, 0.20, 0.25)$.
ARIMA/SARIMA	Burn-in is 300. AR and MA orders are sampled from $\{0, 1, 2, 3\}$ with probabilities $(0.1, 0.3, 0.4, 0.2)$; if both are zero, p is reset to one. Differencing order $d \in \{0, 1, 2\}$ has probabilities $(0.50, 0.40, 0.10)$. AR and MA coefficients follow $\mathcal{U}(-0.8, 0.8)$ and are rescaled when their ℓ_1 norm exceeds 0.92. Innovation scale follows $\mathrm{LogU}(0.02, 1)$. A seasonal AR term is added with probability 0.35, with lag in $\{4, 7, 12, 24, 52, 96\}$ and coefficient $\mathcal{U}(-0.5, 0.5)$.
Regime-switching OU SDE	The number of regimes is 2, 3, or 4 with probabilities $(0.50, 0.30, 0.20)$. The base mean follows $\mathcal{U}(-2, 2)$; signed regime offsets have magnitudes $\mathcal{U}(0.5, 4)$; mean-reversion rates follow $\mathcal{U}(0.03, 3)$; and diffusion scales follow $\mathcal{U}(0.03, 1.5)$. Transition rows are Dirichlet distributed with diagonal concentration $\mathcal{U}(10, 50)$. A sinusoidal perturbation of the regime mean is enabled with probability 0.4, with amplitude $\mathcal{U}(0, 0.8)$ and period $\mathcal{U}(24, \max(25, L/2))$.
Piecewise level/change point	The segment count is a clipped geometric draw, $\min\{\max(2, G + 1), \max(3, \lfloor L/8 \rfloor)\}$ with $G \sim \mathrm{Geometric}(0.15)$. Segment proportions are Dirichlet with concentration $\mathcal{U}(0.5, 2)$. Level modes uniform, random walk, and clustered have probabilities $(0.35, 0.40, 0.25)$. Transition modes hard, ramp, and sigmoid have probabilities $(0.50, 0.30, 0.20)$; smooth-transition width is $\mathcal{U}(0.01L, 0.05L)$. White noise is added with probability 0.70 and scale $\mathcal{U}(0.02, 0.30)$; an additional trend–seasonality component, scaled by 0.3, is added with probability 0.35.
Spike/event	Baseline probabilities for kernel GP, trend–seasonality, flat noise, and OU are $(0.25, 0.45, 0.15, 0.15)$. The event count is a clipped Poisson draw with rate $\max(1, L/160)$. Event-mode probabilities for point, Gaussian, plateau, shock–recovery, and periodic events are $(0.25, 0.30, 0.20, 0.15, 0.10)$. Signed event amplitudes have magnitude $\mathrm{LogU}(0.8, 6)$. Periodic-event spacing is sampled between 16 and approximately $L/4$; plateau width is sampled between 2 and approximately $L/20$; shock–recovery width is sampled between 4 and approximately $L/12$. Final noise scale follows $\mathcal{U}(0.01, 0.25)$.
Analytic waveform	The number of component waves is uniform in $\{1, 2, 3\}$. Sawtooth, square, and triangle shapes are sampled uniformly; amplitude follows $\mathcal{U}(0.3, 3)$, frequency $\mathcal{U}(1, 50)$ cycles per normalized interval, and phase $\mathcal{U}(0, 1)$ cycles. Square-wave duty cycle follows $\mathcal{U}(0.2, 0.8)$. Amplitude modulation is enabled independently for each component with probability 0.30, with modulation frequency $\mathcal{U}(0.5, 5)$ and depth $\mathcal{U}(0.1, 0.8)$. A linear trend is added with probability 0.30, and Gaussian noise with scale $\mathcal{U}(0.01, 0.30)$ is added with probability 0.70.
Fractional Brownian motion/noise	The Hurst parameter follows $\mathcal{U}(0.1, 0.9)$ and fGn versus fBm is selected uniformly. The spectral exponent is $\beta = \mathrm{clip}(2H - 1, -0.8, 0.8)$. fBm is approximated by cumulatively summing colored-noise increments. The final amplitude follows $\mathrm{LogU}(0.1, 5)$.
GARCH-family volatility	GARCH, GJR-GARCH, and EGARCH are selected uniformly; normal and Student-t innovations are selected uniformly. Student-t degrees of freedom follow $\mathcal{U}(3, 10)$. Zero, constant, and AR(1) means have probabilities $(0.55, 0.20, 0.25)$; the AR coefficient follows $\mathcal{U}(-0.3, 0.3)$. Unconditional volatility follows $\mathrm{LogU}(0.005, 0.5)$ and burn-in is 500. GARCH/GJR persistence follows $\mathcal{U}(0.70, 0.97)$; EGARCH uses $\beta \sim \mathcal{U}(0.70, 0.97)$, $\alpha \sim \mathcal{U}(0.05, 0.25)$, and $\gamma \sim \mathcal{U}(-0.20, 0.20)$. Returns are cumulatively summed with probability 0.5.
Chaotic/audio-inspired	Lorenz, Mackey–Glass, NARMA, and audio-like subtypes are selected with probability 0.25 each. Lorenz uses burn-in 800, step size $\mathcal{U}(0.005, 0.02)$, $\sigma \sim \mathcal{U}(8, 12)$, $\rho \sim \mathcal{U}(24, 30)$, and $\beta \sim \mathcal{U}(2, 3.5)$. Mackey–Glass uses burn-in 1,000, delay 15–30, exponent 8–12, coefficient ranges $\mathcal{U}(0.15, 0.25)$ and $\mathcal{U}(0.05, 0.15)$, and step size $\mathcal{U}(0.5, 2)$. NARMA order is 5–14. Audio-like signals contain 2–5 oscillatory layers with frequency $\mathrm{LogU}(1, 80)$ and amplitude $\mathrm{LogU}(0.1, 1.5)$, optional rhythmic events with probability 0.7, and colored noise with $\beta \sim \mathcal{U}(0.5, 2)$.

D.2.2 RANDOM STRUCTURAL CAUSAL MODEL

The online SCM contains K root nodes and P observed non-root nodes. Every primitive series is assigned bijectively to one root, and every root is forced to influence at least one observed node. Observed nodes may additionally depend on earlier observed nodes, so the graph can be deeper than two computational layers while remaining acyclic. The implementation requires

$$K \leq PM,$$

where M is the maximum number of parents. For a root node j and a non-root node j, respectively,

$$x_j(t) = a_j u_{s_j}(t) + \epsilon_j(t), \qquad x_j(t) = \phi_j\left(\sum_{p \in \mathrm{Pa}(j)} w_{jp} x_p(t)\right) + \epsilon_j(t).$$

Anchor roots are first distributed across observed nodes. The remaining parent slots are sampled without replacement from all earlier nodes in the random topological order. Conditional on the admissible range, the parent count is sampled uniformly. Edge weights are independent standard normal draws.

Table 17: Fixed hyperparameter distributions of the online CauKerV2 SCM.

Component	Distribution or value
Root scale a_j	$\mathrm{LogU}(0.6, 1.8)$
Node noise scale	$\mathrm{LogU}(0.01, 0.15)$
Edge weight w_{jp}	$\mathcal{N}(0, 1)$
Activation family	Uniform over linear, ReLU, sigmoid, sine, modulo, leaky-ReLU
Linear slope	$\mathcal{U}(0.5, 2)$; intercept fixed to zero
Modulo constant	$\mathcal{U}(1, 5)$
Leaky-ReLU slope	$\mathcal{U}(0.01, 0.3)$
Sigmoid numerical clip	Input clipped to $[-40, 40]$
Post-SCM normalization	Independently applied to every observed channel

Robust normalization. Primitive roots and final observed channels are normalized independently. Non-finite values are first replaced by zero. For a sequence $\mathbf{z}$, the transformation subtracts the median, divides by a safe standard deviation, clips the result to $[-8, 8]$, subtracts the post-clipping mean, and divides by the standard deviation again. Standard deviations below 10^{-6} are replaced by one. Before padding, the float32 output is additionally clipped to $[-10^6, 10^6]$.

Implemented but separate Arrow-rooted mode. The code also defines an `ArrowCauKerSCMIterableDataset`, in which Arrow records serve as SCM root nodes. This is a separate experimental path: the reported data pipeline uses direct Arrow sampling for KernelSynth and `OnlineCauKerIterableDataset` for CauKerV2. We therefore do not include Arrow-rooted SCM defaults in the reported corpus configuration.

D.3 EMBEDDING-SPACE VISUALIZATION PROTOCOL

Samples. For the generator-family panels, we draw 100 sequences of length 1,024 from each of the ten primitive families listed in Table 13, sampling each family in isolation with its default hyperparameter distributions and without SCM composition, for a total of 1,000 sequences. For the real-versus-synthetic panel, we draw 1,000 complete CauKerV2 samples, generated by the online pipeline of Appendix D.2 with SCM composition enabled, and 1,000 real sequences obtained by sampling GIFT-Eval-Pretrain+ sub-datasets uniformly, so that large sources do not dominate the projection. Multivariate draws contribute one univariate channel each, matching the channel-independent treatment used during pretraining.

Representation and projection. Each sequence is normalized and resampled to the input length of the Mantis encoder (Feofanov et al., 2026), which is used frozen and without any adaptation to our corpora. The encoder output is reduced to ten dimensions by PCA fitted on the sequences of the corresponding panel, and the ten-dimensional representations are projected to two dimensions

by UMAP (McInnes et al., 2020). The two panels are fitted independently, so coordinates are comparable within a panel but not across panels. Exact settings are $n_{\text{neighbors}} = 15$, $\text{min_dist} = 0.1$, the Euclidean metric, and a fixed random seed.

Scope. This visualization is descriptive. UMAP coordinates do not preserve global distances, the layout depends on the chosen encoder and on the neighborhood parameters, and overlap in the projection is not a measure of distributional equality. The figure therefore supports the qualitative statement that the primitive families cover different regions of a frozen representation space, and that CauKerV2 and real data are neither identical nor disjoint in that space; it does not by itself justify the sampling ratios of Appendix D.2.

E Additional Analysis of Deep Quantile Supervision

E.1 Layer-Wise Functional Ablation

We compare the layer-wise sensitivity of checkpoints trained with and without DQS. This experiment uses the smaller model trained for the ablation study, with

$$M = 12, \qquad d_{\text{model}} = 512, \qquad d_{\text{head}} = 32, \qquad P = 16, \qquad T = 8192,$$

an MLP expansion ratio of 4, dropout of 0.1, and 99 output quantiles. For each zero-indexed layer $\ell \in \{0, \ldots, 11\}$, we bypass only that Transformer block during inference by replacing its transformation with the identity mapping,

$$h_{\ell+1} = h_\ell.$$

All other blocks and parameters remain unchanged, and no fine-tuning is performed after the intervention. This measures the sensitivity of the trained predictor to removing one block at a time.

For each checkpoint, we compare every ablation with its corresponding intact model on a fixed set of GIFT-Eval configurations. For both checkpoints, the intact model and all twelve single-block ablations are evaluated on the same full set of 97 GIFT-Eval configurations. In the without-DQS evaluation, the intact model and all twelve ablations cover exactly the same configurations. We report relative changes within each checkpoint.

Let $q \in \{\text{DQS}, \text{noDQS}\}$ denote the training variant, $\mathcal{D}_q$ its evaluation set, and $N_q = |\mathcal{D}_q| = 97$. For metric m, define the relative change caused by bypassing layer ℓ as

$$\Delta^m_{\ell,q} = 100 \left\{ \exp\left[\frac{1}{N_q} \sum_{d \in \mathcal{D}_q} \log\left(\frac{m_{q,\ell,d}}{m_{q,\text{full},d}} \right) \right] - 1 \right\}.$$

Here, $m_{q,\text{full},d}$ is the error of the intact checkpoint and $m_{q,\ell,d}$ is the error after bypassing layer ℓ. Positive values indicate degradation, and negative values indicate improvement. On a fixed evaluation set, this expression also equals the relative change in the Seasonal-Naive-normalized geometric mean, because the normalization factors cancel between the ablated and intact models.

Table 18 shows that the first block has the largest ablation effect in both checkpoints. Bypassing it increases MASE and CRPS by 25.31% and 24.02% with DQS, compared with 20.94% and 20.14% without DQS. In the DQS checkpoint, bypassing any individual block increases both aggregate metrics, with minimum increases of 1.13% in MASE and 0.91% in CRPS.

The clearest qualitative difference occurs at layers 6 and 7. Without DQS, bypassing these layers changes MASE by only $+0.11\%$ and $+0.09\%$, while CRPS improves slightly by 0.05% and 0.14%. With DQS, the corresponding MASE increases are 1.13% and 1.52%, and the CRPS increases are 0.91% and 1.07%. These observations are consistent with DQS reducing local functional redundancy at intermediate depth. The effect is layer dependent: for example, bypassing layer 1 causes greater degradation in the without-DQS checkpoint.

Low aggregate sensitivity does not imply that a block is irrelevant to every task. Without DQS, bypassing layer 6 reduces MASE by 17.11% on `solar/10T/long`, but increases it by 13.79% on `jena_weather/D/short`. Thus, the near-zero aggregate change also reflects offsetting improvements and degradations across tasks. The results therefore support the interpretation that DQS

Table 18: Layer-wise functional ablation on GIFT-Eval with and without DQS. Values are percentage changes relative to each checkpoint's intact baseline; positive values indicate degradation. Each row bypasses one block without fine-tuning. Both training variants cover 97 configurations.

	ΔMASE (%)		ΔCRPS (%)	
Skipped layer	w/ DQS	w/o DQS	w/ DQS	w/o DQS
0	+25.31	+20.94	+24.02	+20.14
1	+7.67	+9.93	+7.54	+9.02
2	+5.86	+4.41	+4.82	+4.57
3	+1.82	+1.69	+2.20	+1.49
4	+1.31	+1.34	+1.54	+1.47
5	+2.50	+2.49	+2.10	+1.69
6	+1.13	+0.11	+0.91	-0.05
7	+1.52	+0.09	+1.07	-0.14
8	+1.63	+0.93	+1.45	+0.92
9	+1.39	+1.64	+1.57	+1.59
10	+1.52	+1.66	+1.42	+1.00
11	+3.08	+2.69	+2.94	+2.10

encourages useful transformations at intermediate depth. This evidence concerns the tested checkpoints and single-block interventions; it does not establish parameter inactivity or the effect of jointly removing multiple blocks.

F Zero-Shot Anomaly Detection: Details

Setup We evaluate on TSB-AD-U (Liu & Paparrizos, 2024) using the benchmark's unsupervised protocol: each detector receives the full series and no training data. The per-series sliding window for VUS-PR is set by the benchmark's own `find_length_rank` convention (rank 1). We report mean VUS-PR over the 350 series of the evaluation split; the latent distance metric was selected on the disjoint 48-series tuning split, and no selection of any kind was performed on the evaluation split.

Scoring Each series is tiled into non-overlapping windows of the model's context length W, with the final window right-aligned to cover the tail. Series shorter than W form a single left-padded window; series shorter than one patch fall back to a z-score. Each window is normalised over its visible points as $\tilde{x} = \operatorname{asinh}\big((x-\mu_w)/\sigma_w\big)$, making the score invariant to per-window level and scale. Windows are encoded by the frozen backbone and we take the output of the final Transformer block; the quantile head is discarded. Padding patches are dropped and the remaining content patches are pooled across all windows into $\mathbf{P} \in \mathbb{R}^{n_p \times d}$, with $n_p \approx T/L$ for patch size L. We fit $\boldsymbol{\mu} = \operatorname{mean}(\mathbf{P})$ and the sample covariance $\boldsymbol{\Sigma}$, apply shrinkage $\boldsymbol{\Sigma}_\lambda = (1-\lambda)\boldsymbol{\Sigma} + \lambda \frac{\operatorname{tr}(\boldsymbol{\Sigma})}{d}\mathbf{I}$ with $\lambda = 0.1$, and score each patch by $a_i = \sqrt{\max\big((\mathbf{p}_i - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}_\lambda^{+} (\mathbf{p}_i - \boldsymbol{\mu}), 0\big)}$ using the pseudo-inverse. Patch scores are broadcast to the L timesteps they cover, averaged where windows overlap, and min–max scaled to $[0, 1]$.

Metric selection and results Table 19 compares latent distance metrics on the tuning split. Full-covariance Mahalanobis is best on both mean and median, and its margin over the diagonal variant indicates that off-diagonal structure in the embedding distribution carries usable signal. The selected metric transfers without degradation: 0.4379 mean VUS-PR on tuning against 0.4282 on evaluation, with the median improving from 0.3716 to 0.4024 on the larger split. Table 20 reports the full set of evaluation-split measures, with all 350 series scored successfully. Scoring requires $\lceil T/W \rceil$ forward passes per series.

Table 19: Latent distance metrics on the TSB-AD-U tuning split (48 series).

Metric	Mean	Median
Mahalanobis (full)	**0.4379**	**0.3716**
Centroid (robust)	0.4077	0.3190
Mahalanobis (diagonal)	0.4075	0.3366
Centroid	0.3713	0.2669

Table 20: TABBY-PRETRAIN on the TSB-AD-U evaluation split (350 series).

Measure	Mean	Median
VUS-PR	0.4282	0.4024
AUC-PR	0.3408	0.2643
VUS-ROC	0.8364	0.9334

G OPTIMIZER-AWARE ANALYSIS OF THE DQS UPDATE DIRECTION

G.1 QUESTION AND GRADIENT CONSTRUCTION

Let θ_0 denote the parameters of the 110K TABBY-SMALL checkpoint. Let $\mathcal{L}_T$ be the final-exit quantile pinball loss and $\mathcal{L}_D$ the unscaled DQS loss. For a 12-block backbone, the latter is

$$\mathcal{L}_D = \sum_{k\in\{3,6,9,12\}:\, k<12} \left(\frac{k}{12}\right)^{\gamma} \mathcal{L}^{(k)}_{\text{pinball}}, \qquad \gamma = 1. \tag{11}$$

The deepest exit is the final prediction objective and is consequently not counted again as auxiliary supervision. On one shared update batch, we compute

$$g_T = \nabla_\theta \mathcal{L}_T(\theta_0), \qquad g_D = \nabla_\theta \mathcal{L}_D(\theta_0). \tag{12}$$

We then construct

$$g_A = g_T, \tag{13}$$

$$g_B = g_T + \lambda_{\text{DS}} g_D, \qquad \lambda_{\text{DS}} = 0.5, \tag{14}$$

$$g_M = c g_T, \qquad c = \frac{\|g_B\|_2}{\|g_T\|_2}. \tag{15}$$

Thus, g_M and g_B have the same Euclidean norm, while g_M retains the direction of g_T. Comparing B with A alone would not separate a directional effect from an increase in gradient magnitude; B versus M provides this control.

G.2 OPTIMIZER-AWARE EQUAL-DISPLACEMENT INTERVENTION

AdamW does not map a current gradient to a parameter displacement by a simple scalar multiplication because the stored first- and second-moment states affect every coordinate. We therefore retain the optimizer state (m_0, v_0) from the checkpoint and pass all three gradients through exactly the same optimizer transformation. Denoting the resulting parameter displacement by

$$u_s = \text{AdamWStep}(\theta_0, g_s; m_0, v_0) - \theta_0, \qquad s \in \{A, M, B\}, \tag{16}$$

we construct the equal-radius displacement

$$\widetilde{u}_s(\rho) = \rho \|u_A\|_2 \frac{u_s}{\|u_s\|_2}, \qquad \rho \in \{0.25, 0.5, 1\}. \tag{17}$$

Consequently,

$$\|\widetilde{u}_A(\rho)\|_2 = \|\widetilde{u}_M(\rho)\|_2 = \|\widetilde{u}_B(\rho)\|_2. \tag{18}$$

Weight decay is set to zero for all branches so that the intervention isolates task-gradient effects, while the checkpoint learning rate and AdamW moments are retained. The same clipping rule is used for every branch. Only one of the 92 trials triggers clipping for M and B; their clipping factors are equal in that trial. The maximum relative mismatch between the final displacement norms is 2.61×10^{-5}.

For each branch, we temporarily evaluate

$$R_s(\rho) = R_V(\theta_0 + \widetilde{u}_s(\rho)), \tag{19}$$

where R_V is the final-exit pinball loss on a held-out validation batch. The model and optimizer are restored after every branch and every trial. Exact restoration is verified: the repeated baseline loss has zero numerical difference, the model parameters are restored exactly, and the optimizer state hash is unchanged.

We report

$$\Delta_{BA}(\rho) = R_A(\rho) - R_B(\rho), \tag{20}$$

$$\Delta_{BM}(\rho) = R_M(\rho) - R_B(\rho), \tag{21}$$

$$G_B(\rho) = R_0 - R_B(\rho), \tag{22}$$

where $R_0 = R_V(\theta_0)$. Positive Δ_{BA} and Δ_{BM} mean that B has lower validation loss than the corresponding control. Positive G_B means that B also improves upon making no update. The relative effects divide each paired difference by R_0 before aggregation.

The validation gradient

$$g_V = \nabla_\theta R_V(\theta_0) \tag{23}$$

is used only for diagnosis and never to construct A, M, or B. For a sufficiently small displacement,

$$R_V(\theta_0 + u) = R_V(\theta_0) + \langle g_V, u \rangle + o(\|u\|_2), \tag{24}$$

which motivates measuring alignment with the validation descent direction. The final conclusions nevertheless use the actually evaluated losses rather than only this first-order approximation.

G.3 Data and Evaluation Protocol

We use only the official training portion of GIFT-Eval; benchmark test targets are never used. Configurations without a sufficiently long observed training series are excluded. The retained set contains the following 23 configurations from 16 original dataset groups. Their retained configurations are summarized below; "base" denotes the repository's unqualified configuration.

Dataset group	Configuration(s)	Dataset group	Configuration(s)
LOOP Seattle	5T, H	ETT1	15T, H
M-DENSE	H	ETT2	15T, H
Bitbrains Fast	5T	Jena Weather	base, 10T, H
Bitbrains RND	5T	KDD 2018	H
BizITObs App.	base	M4 Daily	base
BizITObs L2C	5T	SaugeenDay	D
BizITObs Service	base	Solar	10T, H
Electricity	15T, H	US Births	D

For each retained series, the official training history is split temporally at 70%. Update windows are sampled from the earlier region and validation windows from the later region, separated by a gap of 128 time steps. Each window has 1,024 input time steps and a 32-step terminal forecasting target. At most 32 series are retained per configuration, and at least 95% of each window must be observed. Normalization statistics are computed only from the visible context in the asinh-normalized loss space.

Each trial uses one update window from every configuration, giving an update batch size of 23. The validation configuration is rotated across trials, with four trials per configuration and 92 paired trials in total. The prediction head is frozen, but gradients still propagate through it into the shared backbone. The trajectory regularizer is disabled, ensuring that B differs from A only through DQS.

G.4 Aggregation and Uncertainty

Multiple frequencies and repeated windows from the same original dataset are not treated as independent observations. For every metric, we first average the paired trial-level effects within each of the 16 original dataset groups and then take the unweighted macro-average across groups. We obtain 95% confidence intervals by resampling these original-dataset group means with replacement for 10,000 paired cluster-bootstrap replicates. The fraction of positive trials is reported descriptively only; the effective unit for the reported interval is the original dataset group, not the individual window. This analysis concerns one fixed checkpoint and optimizer history and does not represent variation across independent pretraining runs.

Table 21: Raw one-step loss differences, in units of 10^{-6}. Positive values favor B. Parentheses give 95% paired original-dataset cluster-bootstrap confidence intervals.

ρ	Δ_{BA}	Δ_{BM}	G_B
0.25	4.04 [0.43, 8.07]	4.14 [0.72, 7.94]	14.74 [0.06, 33.49]
0.50	8.05 [1.07, 15.97]	8.25 [1.51, 15.72]	28.83 [−0.09, 65.58]
1.00	15.99 [1.74, 31.48]	16.37 [2.98, 30.96]	55.19 [−0.52, 124.86]

Table 22: Relative one-step advantages of B at the primary radius $\rho = 1$, in percent. Positive values favor B. Results are first averaged over trials and configurations belonging to the same original dataset.

Dataset group	B–A	B–M	Dataset group	B–A	B–M
LOOP Seattle	0.0136	0.0112	ETT1	0.0087	0.0085
M-DENSE	0.0365	0.0371	ETT2	−0.0210	−0.0200
Bitbrains Fast	−0.0190	−0.0125	Jena Weather	0.0138	0.0148
Bitbrains RND	0.0217	0.0188	KDD 2018	0.0023	0.0047
BizITObs Application	0.1843	0.1578	M4 Daily	−0.0237	−0.0246
BizITObs L2C	0.1056	0.0990	SaugeenDay	0.0421	0.0431
BizITObs Service	0.0282	0.0275	Solar	0.0489	0.0383
Electricity	−0.0149	−0.0145	US Births	0.0046	0.0070

G.5 Complete Aggregate Results

Table 21 reports the corresponding effects in raw pinball loss units. B is better than both A and M at every radius. The B–M confidence interval remains above zero throughout.

The point estimate $G_B > 0$ suggests that B can also be an absolute descent step for the held-out objective. However, its interval narrowly includes zero at the pre-specified primary radius $\rho = 1$, and the relative-gain interval also includes zero. We therefore use Δ_{BM}, not G_B, as the evidence for the stated directional claim.

The three radii probe the same directions and must not be counted as three independent replications. Nevertheless, they provide a useful locality check: all 16 dataset groups retain the same B–M sign across radii, the median ratio between the effects at $\rho = 1$ and $\rho = 0.25$ is 3.98, and the correlation between the first-order prediction and the measured B–M effect is 0.997 at $\rho = 1$.

G.6 Results by Original Dataset Group

The B–M group mean is positive for 12 of 16 groups. The macro-average is not determined by one favorable dataset: leave-one-group-out analysis keeps the B–M effect positive in every case, with a minimum remaining relative effect of 0.0159%. The median group effect is 0.0130%, compared with the mean of 0.0248%, indicating heterogeneous but not purely outlier-driven gains.

G.7 Directional and Layerwise Diagnostics

The average gradient-norm ratio is

$$\frac{\|g_B\|_2}{\|g_T\|_2} = 1.432, \tag{25}$$

while

$$\cos(g_T, g_D) = 0.717. \tag{26}$$

The latter corresponds to an angle of approximately 44°, confirming that DQS is not merely a positive scalar multiple of the final-loss gradient. The AdamW proposal cosines are 0.9946 between A and M and 0.9616 between A and B, corresponding to average angles of approximately 5.8° and 15.9°, respectively. Thus, gradient scaling interacts slightly with the stored optimizer state, which motivates retaining M, but DQS produces a substantially larger directional change.

For layer j, we additionally compute

$$A_j = \cos(g_{B,j}, g_{V,j}) - \cos(g_{T,j}, g_{V,j}). \tag{27}$$

Table 23: Layerwise change in gradient alignment, A_j. Parentheses give 95% original-dataset cluster-bootstrap intervals. These correlated layerwise analyses are exploratory and are not corrected for multiple comparisons.

Parameter group	Alignment change	Parameter group	Alignment change
Position embedding	$0.00039\ [-0.00096, 0.00169]$	Block 7	$0.00079\ [-0.00060, 0.00226]$
Block 1	$0.00197\ [-0.00015, 0.00423]$	Block 8	$0.00071\ [-0.00060, 0.00204]$
Block 2	$0.00178\ [-0.00144, 0.00519]$	Block 9	$0.00193\ [-0.00022, 0.00415]$
Block 3	$0.00218\ [-0.00036, 0.00455]$	Block 10	0
Block 4	$0.00208\ [-0.00103, 0.00507]$	Block 11	0
Block 5	$0.00145\ [-0.00190, 0.00448]$	Block 12	0
Block 6	$0.00045\ [-0.00267, 0.00356]$	Input projection	$\mathbf{0.01051}\ [0.00115, 0.02077]$

Table 23 reports this exploratory diagnostic.

All 11 parameter groups reached by the auxiliary exits have a positive mean alignment change. The input projection is the only individual group whose uncorrected interval excludes zero. Blocks 10–12 receive no auxiliary DQS gradient by construction, so their alignment change is exactly zero. Because the prediction head is frozen, the observed effect must arise through changes to the shared backbone rather than direct adaptation of the readout head.

G.8 Scope of the Conclusion

The experiment establishes a local comparative statement:

> At one fixed 110K checkpoint and optimizer state, and for the sampled GIFT-Eval training-data mixture, DQS produces an optimizer-aware parameter update that is more favorable to the held-out final forecasting objective than both the final-only update and a raw-gradient-norm-matched control.

It does not establish that every DQS step is an absolute validation descent step, that the same effect holds at every stage of pretraining, or that the end-to-end DQS gain is caused exclusively by this mechanism. Establishing those stronger claims would require replication across checkpoints and pretraining seeds, together with controlled multi-step continuation runs.

H Restart-Peak Sensitivity of Stage-End Checkpoints

H.1 Objective

The progressive convergence learning rate schedule reduces the stable learning rate after every completed stable–decay stage. This design is motivated by the hypothesis that a peak learning rate that is appropriate early in pretraining may become excessively disruptive after the model has entered a better-converged region. We therefore examine two related questions:

1. Does the empirically optimal restart peak decrease as pretraining progresses?
2. Does the validation penalty induced by a fixed aggressive restart increase at later checkpoints?

The first question concerns the location of the optimum, whereas the second concerns the steepness of the learning-rate response curve. These questions must be distinguished because the optimum can remain at the boundary of the search grid even when sensitivity to larger learning rates changes substantially.

H.2 Models and Starting Checkpoints

We conduct the experiment on both TABBY-SMALL and the full TABBY-PRETRAIN model. The model architecture is recovered directly from each checkpoint, and all comparisons are performed only between checkpoints of the same model. Absolute validation losses are not compared across model sizes.

Table 24: Models and checkpoint branches used in the restart-peak experiment. The continuation batch size is the effective batch size used by the single-GPU probe, rather than the global batch size of the original distributed pretraining run.

Configuration	TABBY-SMALL	TABBY-PRETRAIN
Parameters	40.291M	145.843M
Transformer layers	12	20
Hidden dimension	512	768
Attention-head dimension	32	64
Context length	8,192	8,192
Patch size	16	16
Output quantiles	99	99
Starting checkpoints	55K, 110K	55K, 110K, 165K
Continuation batch size	80	150
GPUs per branch	1	1
Training seeds	1	1

All branches use seed 20260906. For each starting checkpoint, the model parameters, AdamW first- and second-moment estimates, and optimizer step are restored. The learning-rate scheduler is then replaced by the local restart-probe schedule described below. Model architecture, data mixture, masking configuration, weight decay, AdamW coefficients, gradient clipping, deep-supervision objectives, and all other available checkpoint settings are inherited without modification.

The historical checkpoints do not contain the exact per-rank random number generator and data-sampler states. Consequently, the experiment does not reproduce the unobserved data sequence that would have followed the original training run. Instead, all learning-rate branches from the same checkpoint are initialized with the same seed and sampling configuration, providing a controlled common future stream for comparing restart peaks.

H.3 RESTART-PROBE SCHEDULE

For every model checkpoint and candidate peak learning rate, we run 1,500 additional optimizer steps. Let τ denote the local step following the restart, with $\tau = 0$ at the starting checkpoint. The learning-rate schedule is

$$\eta_\tau = \begin{cases} \eta_{\text{peak}}, & 0 \le \tau < T_{\text{stable}}, \\ \eta_{\min} + \dfrac{\eta_{\text{peak}} - \eta_{\min}}{2} \left[1 + \cos\left(\pi u_\tau\right)\right], & T_{\text{stable}} \le \tau < T_{\text{stable}} + T_{\text{cool}}, \end{cases} \tag{28}$$

where

$$u_\tau = \frac{\tau - T_{\text{stable}}}{T_{\text{cool}} - 1}, \qquad T_{\text{stable}} = 1000, \qquad T_{\text{cool}} = 500, \qquad \eta_{\min} = 10^{-5}. \tag{29}$$

The candidate grid is

$$\mathcal{G} = \left\{10^{-5}, 2.5 \times 10^{-5}, 5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}\right\}. \tag{30}$$

No additional warm-up is applied: the learning rate is set directly to η_{peak} at the first continuation step. The $\eta_{\text{peak}} = 10^{-5}$ branch satisfies $\eta_{\text{peak}} = \eta_{\min}$ and therefore serves as a no-restart control with a constant learning rate of 10^{-5}.

The complete experiment contains

$$2 \times 5 + 3 \times 5 = 25 \tag{31}$$

training branches, corresponding to 15,000 continuation steps for TABBY-SMALL and 22,500 continuation steps for the full model.

H.4 FIXED VALIDATION PROBE AND AGGREGATION

Every final branch is evaluated on a fixed pretraining-distribution validation probe containing 256 series from each of the following four sources:

$$\mathcal{D} = \{\text{BLAST}, \text{KernelSynth}, \text{GIFT-Eval-Pretrain+}, \text{CauKerV2}\}. \tag{32}$$

The sampled series and prediction masks are generated once and reused for every model, checkpoint, and learning-rate branch. This probe does not use the downstream GIFT-Eval test set.

For validation series i, let $\mathcal{M}_i$ denote the evaluated masked timestamps and let $\mathcal{Q}$ be the set of 99 quantile levels. The per-series terminal pinball loss is

$$\ell_i = \frac{1}{|\mathcal{M}_i||\mathcal{Q}|} \sum_{t \in \mathcal{M}_i} \sum_{q \in \mathcal{Q}} \rho_q \left(x_{i,t} - \widehat{x}_{i,t}^{(q)} \right), \tag{33}$$

where

$$\rho_q(u) = u \left(q - \mathbb{I}[u < 0] \right). \tag{34}$$

Let $s = (\text{model size}, \text{starting checkpoint})$ index one model–checkpoint pair. Let $n_i = |\mathcal{M}_i|$. Within source d, the losses are aggregated using square-root length weighting:

$$L_{s,d}(\eta) = \frac{\sum_{i \in d} n_i^{1/2} \ell_i}{\sum_{i \in d} n_i^{1/2}}. \tag{35}$$

The scalar score used to compare learning rates is the weighted geometric mean

$$G_s(\eta) = \exp \left[\sum_{d \in \mathcal{D}} w_d \log L_{s,d}(\eta) \right], \tag{36}$$

with

$$w_{\text{BLAST}} = w_{\text{KernelSynth}} = w_{\text{GIFT}} = 0.3, \qquad w_{\text{CauKerV2}} = 0.1. \tag{37}$$

Lower values are better. We quantify restart sensitivity relative to the no-restart control as

$$P_s(\eta) = 100 \left[\frac{G_s(\eta)}{G_s(10^{-5})} - 1 \right]. \tag{38}$$

Thus, $P_s(\eta) > 0$ indicates that increasing the learning rate causes a higher terminal validation loss after the common cooldown.

H.5 COMPLETE LEARNING-RATE RESPONSE

Table 25 reports the weighted geometric validation score for every branch. For every tested checkpoint, the score increases monotonically as the restart peak increases.

The corresponding relative penalties are reported in Table 26.

For TABBY-SMALL, the penalty produced by $\eta_{\text{peak}} = 2 \times 10^{-4}$ increases from 2.487% at 55K to 3.223% at 110K. For the full model, it increases from 2.651% at 55K to 3.596% at 110K and remains nearly unchanged at 3.626% at 165K. The effect therefore develops primarily between 55K and 110K and subsequently approaches a plateau.

Table 25: Complete restart-peak response. Entries are weighted geometric terminal validation losses $G_s(\eta)$; lower is better. Bold values indicate the best candidate for each checkpoint.

Model	Start	10^{-5}	2.5×10^{-5}	5×10^{-5}	10^{-4}	2×10^{-4}
Small	55K	**0.096181**	0.096301	0.096583	0.097285	0.098573
Small	110K	**0.094641**	0.094830	0.095235	0.096137	0.097691
Main	55K	**0.090308**	0.090411	0.090707	0.091437	0.092702
Main	110K	**0.088560**	0.088746	0.089096	0.089913	0.091745
Main	165K	**0.087934**	0.088084	0.088539	0.089310	0.091123

Table 26: Relative validation-loss increase $P_s(\eta)$ over the 10^{-5} continuation. Positive values indicate degradation.

Model	Start	2.5×10^{-5}	5×10^{-5}	10^{-4}	2×10^{-4}
Small	55K	+0.125%	+0.418%	+1.148%	+2.487%
Small	110K	+0.200%	+0.627%	+1.580%	+3.223%
Main	55K	+0.115%	+0.442%	+1.251%	+2.651%
Main	110K	+0.210%	+0.605%	+1.528%	+3.596%
Main	165K	+0.170%	+0.688%	+1.565%	+3.626%

H.6 Response-Curve Sensitivity

To summarize the complete response curve rather than a single aggressive peak, define

$$x(\eta) = \log_2 \frac{\eta}{10^{-5}}, \qquad D_s(\eta) = \log G_s(\eta) - \log G_s(10^{-5}). \tag{39}$$

We fit the descriptive linear response

$$D_s(\eta) = a_s + S_s x(\eta) + \epsilon_{s,\eta} \tag{40}$$

over the five candidate learning rates. The coefficient S_s measures the increase in log validation loss per doubling of the restart peak.

Both models show an approximately 30% increase in response-curve sensitivity from 55K to 110K. For the full model, the additional increase from 110K to 165K is only 1.6%, again indicating that restart sensitivity largely saturates after 110K.

H.7 Source-Wise Sensitivity

Table 28 decomposes the penalty at the most aggressive tested restart, $\eta_{\text{peak}} = 2 \times 10^{-4}$, by validation source.

All four sources become more sensitive between 55K and 110K for the full model. Between 110K and 165K, the BLAST and GIFT-Eval-Pretrain+ penalties continue to increase, whereas the CauKerV2 and KernelSynth penalties decrease moderately. Consequently, the aggregate sensitivity remains approximately constant over the final interval. KernelSynth contributes the largest individual component of the aggregate high-restart penalty, while the contribution from GIFT-Eval-Pretrain+ increases at later checkpoints.

H.8 Boundary Optima and Interpretation

For every checkpoint, the empirical optimum over $\mathcal{G}$ is the smallest candidate:

$$\widehat{\eta}^*_{\text{Small},55K} = \widehat{\eta}^*_{\text{Small},110K} = \widehat{\eta}^*_{\text{Main},55K} = \widehat{\eta}^*_{\text{Main},110K} = \widehat{\eta}^*_{\text{Main},165K} = 10^{-5}. \tag{41}$$

Table 27: Descriptive restart sensitivity measured over the complete learning-rate grid. Larger S_s indicates a steeper validation-loss response to increasing the restart peak.

Model	Start	S_s	Change from previous stage
Small	55K	0.005449	–
Small	110K	0.007098	+30.2%
Main	55K	0.005845	–
Main	110K	0.007694	+31.6%
Main	165K	0.007820	+1.6%

Table 28: Source-wise validation-loss increase under $\eta_{\text{peak}} = 2 \times 10^{-4}$ relative to the 10^{-5} continuation.

Model	Start	BLAST	CauKerV2	GIFT	KernelSynth
Small	55K	+0.922%	+1.885%	+0.959%	+5.859%
Small	110K	+1.302%	+2.234%	+0.973%	+7.870%
Main	55K	+0.882%	+1.220%	+1.980%	+5.632%
Main	110K	+1.548%	+1.925%	+2.766%	+7.117%
Main	165K	+1.690%	+1.492%	+3.360%	+6.608%

A paired, source-stratified bootstrap with 10,000 resamples also places the optimum at 10^{-5} for every checkpoint. However, this produces a boundary solution rather than an interior estimate. The degenerate bootstrap intervals $[10^{-5}, 10^{-5}]$ should therefore not be interpreted as proving that the unconstrained optimal learning rates are exactly equal.

The experiment does not support the strict hypothesis that the location of the optimal restart peak decreases between adjacent checkpoints. It instead supports a different and complementary observation: the validation-loss response to a fixed high restart peak becomes substantially steeper from 55K to 110K and remains elevated at 165K. This distinction is important because the motivation for decreasing restart peaks does not require claiming that the global optimum has been identified by this short-horizon probe.

H.9 LIMITATIONS

The experiment evaluates a short continuation horizon rather than a complete stable–decay stage. A high restart peak may require more than 1,500 updates to recover from its initial perturbation, and the present results therefore characterize short-horizon restart sensitivity rather than the globally optimal peak for a full training stage.

Moreover, each branch is run with one training seed. The paired bootstrap measures uncertainty arising from the finite validation probe, but it does not account for variation across future training streams. Finally, the single-GPU continuation batches differ between TABBY-SMALL and the full model. The two models are therefore used as independent replications of the stage-wise trend; their absolute sensitivity coefficients should not be directly compared.