THE FORECASTING COMPANY

t_0: A Time-Series Foundation Model for Forecasting with Context

Lucas Meyer* **Claudio Sole*** **Huikan Xiang*** **Nicolas Li** **Lucas Franceschino**
Arnau Quera-Bofarull§ **Joachim Fainberg*** **Geoffrey Négiar***

*Equal contribution. §Macrocosm, author of the independent evaluation in Section 5.6.2.
Correspondence: research@theforecastingcompany.com

We present `t0-alpha`, an open-weights 102M-parameter foundation model for forecasting with multivariate context. The model conditions its forecasts on target history, past covariates, and known-future covariates, without task-specific retraining. Its transformer layers alternate attention along time and across variates, and it produces probabilistic forecasts through quantile predictions. Pretraining combines curated public data with synthetic generator families constructed to contain covariate-to-target dependencies. On GIFT-Eval, `t0-alpha` achieves an aggregate CRPS of 0.4941. On fev-bench, it achieves a skill score of 42.2. Supplying known-future covariates improves skill by 6.3 percentage points across 30 tasks. It reduces scaled quantile loss of the predictions by 51% on German day-ahead electricity price predictions and by 43% on daily Rossmann sales. We also report results for `t0-beta`, a 256M-parameter successor released with this report, which places among the strongest zero-shot models compared here. It achieves a CRPS of 0.4738 and a MASE of 0.6865 on GIFT-Eval, third on both metrics and within 4.0% of the best zero-shot TSFM. On fev-bench it obtains a skill score of 46.4, ranking third again and 2.3 percentage points behind the leader. These results make `t0-beta` one of the best open-weights models with multivariate support released to date.

Date: September 17, 2026
Weights (`t0-alpha`): huggingface.co/theforecastingcompany/t0-alpha
Weights (`t0-beta`): huggingface.co/theforecastingcompany/t0-beta
Code: https://github.com/theforecastingcompany/tfc-t0
Python Package: `tfc-t0`

TABLE OF CONTENTS

1 Introduction

2 Problem Formulation

3 Related Work

4 The t_0 Model
- 4.1 Architecture
- 4.2 Training
- 4.3 Inference

5 Evaluation
- 5.1 Public benchmarks
- 5.2 Model capability analysis
- 5.3 Robustness to missing data
- 5.4 Context efficiency
- 5.5 Rollout strategy
- 5.6 Use cases

6 Limitations

7 Conclusion

1 Introduction

Organizations plan against the future they expect. How much energy a region will consume next week, how many data centers to build over five years, whether demand for winter clothing rises next season: energy management, capacity provisioning, inventory management, and staffing all rest on a forecast.

Forecasting methods have moved from simple baselines to statistical models, then to deep learning, and most recently from task-specialized models to general time series foundation models (TSFMs). The *naive* forecast repeats the last observed value across the whole horizon, and a seasonal naive forecast repeats the value observed one seasonal period earlier. Above them sit models fitted to data. ARIMA [11] and gradient-boosted trees such as XGBoost [13], CatBoost [57], and LightGBM [34] took the top places in the 2020 M5 forecasting competition [45] and were until recently the *de facto* choice. Every model of that generation is fitted anew for each dataset, and refitted as the distribution shifts, with manual tuning and feature engineering at each turn.

Time series foundation models such as Chronos [5], Moirai [72], Toto [15], and `t0-alpha`, the subject of this report, can forecast a dataset unseen at training time, thus achieving so-called zero-shot forecasting. Pretrained once on a large and diverse corpus, they generalize at inference time as large language models do, and the best of them now beat tuned task-specific baselines on public benchmarks [2]. Most of them, however, read one univariate series and nothing else. Forecasting problems rarely come that bare. Demand responds to promotions, holidays, the weather and local events, and the items of a catalog move together (Figure 1). We built `t0-alpha` to forecast conditioned on that context.

This report also introduces `t0-beta`, its open-weights 256M-parameter successor, which places third on GIFT-Eval and third on fev-bench among the zero-shot models we compare, and carries its largest margins on the tasks that come with covariates.

Our contributions are as follows.

1. **An open-weights artifact.** A 102M-parameter TSFM released on Hugging Face, with an inference Python package (`tfc-t0`) and a notebook that reproduces our GIFT-Eval results.
2. **An architecture that leverages covariates through alternating attention.** Besides relying on attention along the time axis, the model attends across variates, which carry the past and known-future covariates alongside the target.
3. **A thorough evaluation of covariate lift and deployment properties.** On the fev-bench public benchmark, passing covariates to the same model checkpoint raises skill by 6.3 points on the tasks with known-future covariates and by 2.7 points on those with past covariates only. The gains concentrate where a covariate plausibly drives the target. Beyond accuracy on the public benchmarks, we evaluate and report the properties a TSFM needs for deployment on real applications. We validate the calibration of `t0-alpha` and the behavior of its tails, measure its resilience to gaps in the context history, study how its accuracy holds as the context shortens, check that its rank is stable across the eleven metrics the leaderboard records, and benchmark it on an electricity-demand case study.

2 Problem Formulation

A *time series* is a collection of related variates indexed by the same time axis, where a *variate* is a quantity observed at successive time steps, $y_{1:T} = (y_1, \ldots, y_T)$ with $y_t \in \mathbb{R}$ for every t. When a time series has a single variate it is said to be univariate. The time series is then synonymous with its variate. Real-world applications

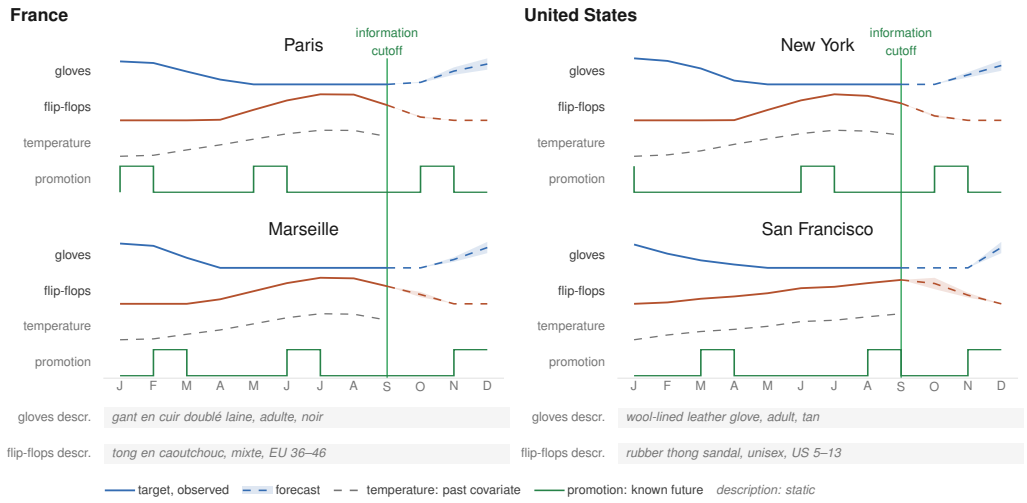

Figure 1. Illustrative example of a time series problem involving multiple covariates. Demand must be forecast for two retail items in stores across different cities and countries, using historical temperatures, known-future promotions, and the product description.

typically involve multivariate time series. For instance, a retailer might forecast the demand for seasonal items based on temperature and promotions across different cities (Figure 1).

The variates of a time series differ in the role they play, and in when their values become available. A time series holds $V \geq 1$ *target* variates $\mathbf{y} = (y_{1:T}^{(v)})_{v=1}^{V}$, the quantities to forecast. It may also hold $P \geq 0$ *past covariates* $\mathbf{x} = (x_{1:T}^{(p)})_{p=1}^{P}$, observed over the context $1{:}T$ only, and $F \geq 0$ *known-future covariates* $\mathbf{z} = (z_{1:T+H}^{(f)})_{f=1}^{F}$, observed over the context and over a horizon of H steps. Covariates differ from target variates in that they carry information the forecast can use but are not themselves predicted for the task at hand. Figure 1 illustrates this distinction as the task is to predict item demands with the help of weather information, and not predicting the weather itself. A time series may include *static covariates* as well, which do not vary in time, such as the category of a product or the format of a store. We leave static covariates to future work.

Forecasting starts from an *information cutoff datetime*, the time step T that bounds the context and after which nothing is observed, except for future covariates. Given a prediction length of H steps, the task is to predict the targets over $T{+}1, \ldots, T{+}H$ from the context, that is everything observed up to T, together with any known-future covariates over the prediction length itself.

A model can answer that task with a single value per time step, but this is rarely enough. Most applications require predictions with associated probabilities. For instance, staffing and energy-trading decisions hinge on how costly the worst plausible outcomes are, not on the expected outcome alone. Models are thus expected to predict multiple values for each time step, corresponding either to different samples or to quantile levels directly.

Given a set of $Q = |\mathcal{Q}|$ quantile levels $\mathcal{Q} \subset (0, 1)$, a TSFM produces a forecast $\hat{y}_{T+h}^{(v),q} \in \mathbb{R}^{V \times H \times Q}$, an estimate of the q-quantile of the marginal distribution of $y_{T+h}^{(v)}$ conditional on everything observed (Equation 1).

$$\hat{y}_{T+h}^{(v),q} = f_\theta(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}, \mathbf{z}_{1:T+H})_h^{(v),q}, \qquad v \in [V],\ q \in \mathcal{Q},\ h \in [H]. \tag{1}$$

The goal of a TSFM, f_θ, is to predict $\hat{y}$ for *any* time series, regardless of V, P, F, T, and H - that is, regardless of how many targets and covariates it carries, how much context it provides, and how long a horizon is required.

3 Related Work[1]

Traditional statistical methods Forecasting vastly predates deep learning. The roots of modern forecasting may be found in antiquity, when Babylonian scholars turned centuries of recorded observations into omen-based inference and, by the first millennium BCE, eclipse prediction [61, 65]. As such, forecasting methods have evolved throughout the years [32]. Three families account for most of the practice that came before foundation models: the autoregressive integrated moving average (ARIMA) family [11], exponential smoothing, and gradient-boosted decision trees (GBTs). Each of them fits its parameters to one series or one dataset at a time.

As its name suggests, ARIMA combines three mechanisms. It is *autoregressive*: the next value is regressed on the values that precede it. It takes a *moving average*: the error term of that regression is itself a linear combination of past errors. And it is *integrated*: both apply to the differences between consecutive time steps rather than to the raw series, a transformation that removes trend. Each mechanism carries an order, which dictates the number of associated parameters that are fitted at training time. Variants extend the same core with seasonal terms (SARIMA [12]), exogenous regressors (ARIMAX [51]), and a vector form for several series at once (VARIMA [44]).

Exponential smoothing methods produce forecasts by averaging the history of a time series, modulating each observation by a factor that shrinks exponentially with time. They are described by the ETS framework, whose name comes from the three components it considers: error, trend, and seasonality [33]. Every component carries parameters of its own.

The third family of forecasting methods, the GBTs, originates from tabular machine learning rather than from the time series domain [24]. Nonetheless, they have been used successfully in most of the top-ranked entries of the M5 forecasting competition [45]. Forecasting with them requires flattening the series into a table of lagged values, calendar features, and covariates, which is how they absorb promotions or weather with no change of machinery. The most common implementations, namely XGBoost [13], LightGBM [34], and CatBoost [57], differ in how each tree is fitted and how categorical features are handled.

Deep learning models While ARIMA and ETS methods are limited to a few parameters fitted per time series, deep learning methods fit many more parameters on datasets of time series, following a trend initiated by GBTs. They also relax the assumptions made about the structure of the forecast. Nothing prescribes a decomposition into level, trend, and seasonality. Which components matter and how they combine is now left to the model to learn from the data.

The models that followed diverge on three axes: how a series enters the network, the architecture that processes it, and the form of the output. On the first, DeepAR [62] and MQ-RNN [71] read the series one step at a time through a recurrence. N-BEATS takes the whole lookback window at once, as a flat vector [50]. PatchTST cuts it into patches of consecutive steps and treats each patch as a token, shortening the sequence attention must cover [47]. On the second axis, the backbone is a recurrent network for DeepAR and MQ-RNN, a deep stack of fully connected layers tied by backward and forward residual links for N-BEATS, and attention for PatchTST and MQTransformer [23]. They differ last in the form of the output. N-BEATS and PatchTST are point-forecasters. They produce one value per horizon step, without any information about its uncertainty. DeepAR instead emits the parameters of a distribution at every step, and forecasts are drawn from it as sample paths. The MQ family drops the sampling altogether and decodes every horizon step and every quantile level in a single pass.

For all their differences, the models of this generation share the same constraints: each is fitted to a single dataset and generalizes poorly beyond it. Each is also tied to a single input schema. N-BEATS and PatchTST

[1] An interactive version of this section is available as a blog post: From ARIMA to Foundation Models.

only handle univariate time series. The MQ family declares a fixed number of past, future, and static covariates prior to training. A new dataset, or a different number of covariates, requires retraining.

Time series foundation models Foundation models supersede the previous generations through their versatility. Where prior models had to be fitted anew for every dataset, a single TSFM forecasts series from datasets it has never seen. No retraining is required.

Their architectures vary, but along recurring axes. The first is about how a model ingests a time series. Most TSFMs break the series into patches. Consecutive steps are grouped together, and the model produces an embedding for each group. A patch is the closest analogue to the token of a language model, with one major difference: language tokens are discrete, whereas time series typically hold continuous values. Moirai varies the patch size with the frequency of the series [72]. Chronos-2 and Toto keep a single patch size, which is now the common practice [6, 15]. Other TSFMs, like TS-ICL and Falcon-X, abandon the patch grid completely to avoid cutting the series artificially: the former casts forecasting as timestamp-aligned regression [39], the latter maps each variate into a shared latent space [40]. Patches reappear on the output side, where one decoded patch covers several horizon steps. Toto-2 produces multiple patches from the same context before relying on rollout to generate forecasts beyond a given horizon [6]. TimesFM-3.0 decodes horizon patches longer than its stride and blends the overlap [28], while forking sequences decode a forecast from every position of the series in a single pass, so that the overlapping horizons average part of the volatility away [56, 71].

Regardless of how a model embeds the series, the result is always a sequence of tokens. The second axis is how these tokens exchange information. Most TSFMs are transformers whose attention runs along the time axis [6, 15, 16]. The TiRex line is the exception, mixing along time with xLSTM blocks instead [8, 9, 54].

Most TSFMs are univariate. They take one variate per series and run attention along the time axis only. We distinguish multi-target models, which forecast several targets of one series in a single pass, from multivariate models, which also condition on past or known-future covariates. TSFMs with covariate support add a second attention along the variate axis, decoupled from the first. Chronos-2, TiRex-2, and TimesFM-3.0 all alternate the two [6, 28, 54]. A complementary line of work adds no native covariate support, and instead adapts frozen univariate TSFMs to covariates *post hoc* [7, 17, 53, 59].

TSFMs also differ in the form of their output, though here they appear to be converging. One group predicts the parameters of a distribution and draws sample paths from it. For instance, Toto took this route [15]. Most models now emit quantiles at fixed levels instead. The Chronos family, TiRex-2, and Toto-2, the successor of Toto, all belong to this group [5, 6, 35, 54]. Quantiles are simpler and cheaper, since sampling adds a step at inference, but they are marginal, one per horizon step. Predicting sample paths is still advantageous where the joint structure across steps matters.

How a model is trained follows from these architectural choices. Encoder-only models such as Chronos-2 borrow the architecture of masked-language models, with a slightly different objective. Rather than masking patches at random and reconstructing them, they split each training example into a context and a future window. They then compute the quantile loss on the future window alone [6]. Decoder-only architectures such as Toto-2 rely on teacher forcing and next-patch prediction with causal attention along the time axis [35]. The two types of models blur under the contiguous patch masking (CPM) introduced by TiRex, where spans of patches are blanked at random and the model must fill them in or continue past them [8]. Toto-2 and TimesFM-3.0 both reuse it [28, 35], as does the model introduced in this report.

In contrast to the model architecture choices, pretraining corpora are more uniform across models. Most models mix public datasets, an internal pool for the teams that hold one, and synthetic series. LOTSA [72] and GIFT-Eval Pretrain [2] are the most common public datasets. Toto also trains on internal observability telemetry [15].

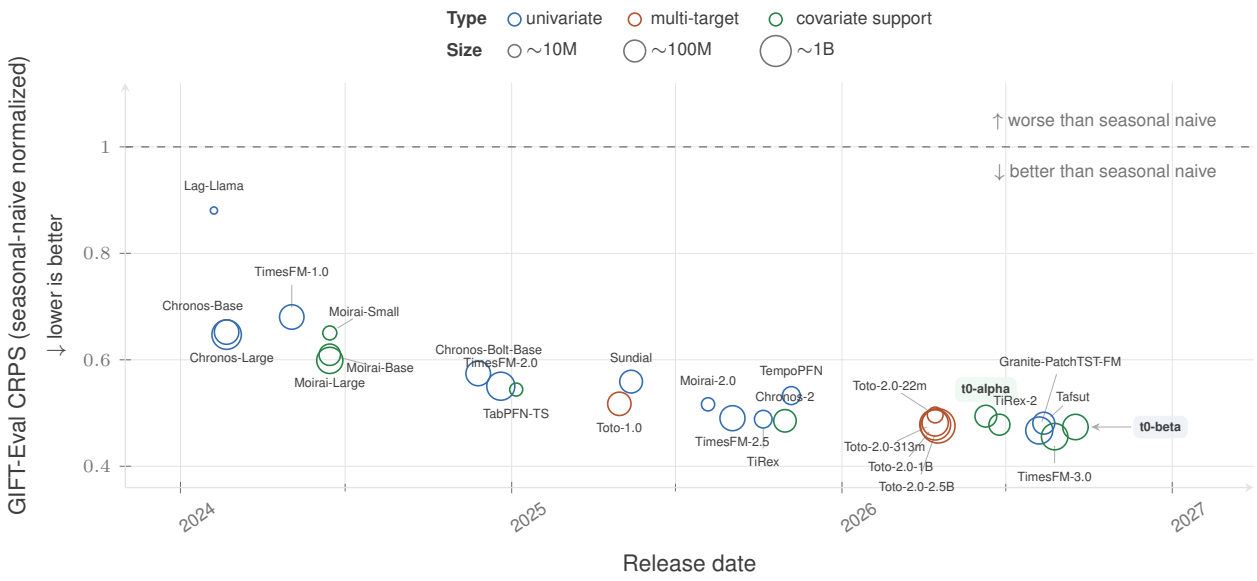

Figure 2. The TSFM landscape on the GIFT-Eval benchmark [2]. Each model sits at its release date against its aggregate CRPS, normalized by the seasonal-naive baseline. Ring radius represents the parameter count on a log scale, from the 2.4M of Lag-Llama to the 2.45B of Toto-2.0-2.5B. Color informs the variate capability of the model: univariate, one variate per series; multi-target, several variates of one series forecast jointly in a single pass; or covariate support, covariates leveraged to forecast the target. `t0-alpha` and `t0-beta` are highlighted.

The synthetic component follows the same handful of recipes almost everywhere: Gaussian-process kernel composition [5], causal kernels [73], structural priors, and random transforms. They are then combined in varying mixes by TiRex, TS-ICL, and Chronos-2 [6, 8, 39].

Finally, progress on TSFMs, as elsewhere in machine learning, is paced by benchmarks. The most widely adopted are GIFT-Eval [2] and fev-bench [64]. The series of GIFT-Eval are mostly univariate and leave covariates little if any role. The more recent fev-bench does include covariate tasks, yet univariate models such as TiRex still sit near the top of its leaderboard. This admits two readings: univariate accuracy comes first, a model that reads covariates but forecasts the target poorly helps no one, and the benchmark does not isolate covariate skill as sharply as it could. The field still has room for benchmarks that target specific capabilities, covariate use among them. Figure 2 places the models of this section on GIFT-Eval, evaluated with the continuous ranked probability score (CRPS), which compares a full predictive distribution against the single observed value.

4 The t_0 Model

We introduce t_0, a model family and its architecture. This report describes `t0-alpha`, the first released model of this family.

4.1 Architecture

t_0 is a decoder-style patch transformer with decoupled time and variate attention. Figure 3 gives an overview of the architecture.

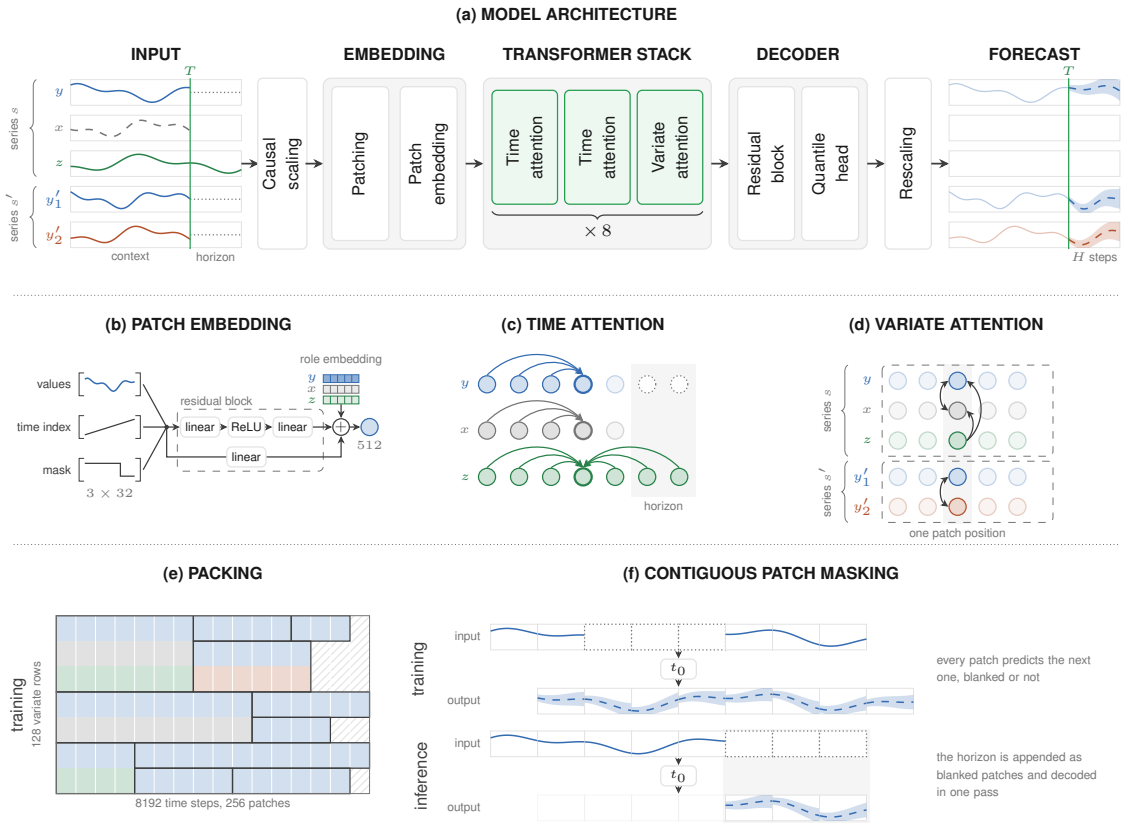

Figure 3. The t_0 pipeline. **(a)** Time series are processed in parallel in one mini-batch. They are normalized with causal statistics, cut into patches, embedded (b), read by a stack alternating time (c) and variate (d) attention blocks, and decoded into quantiles, rescaled at inference into the forecast of the requested targets. **(b)** Embedding of one patch. **(c)** Time self-attention, causal for targets and historical covariates, bidirectional for known-future ones. **(d)** Variate self-attention, restricted to the variates of one time series. It is bidirectional between targets and past covariates, asymmetric for known-future ones, to avoid a two-hop leak of future information. **(e)** Packing of time series of different lengths into one mini-batch. The variate dimension is the batch dimension. Series boundaries are marked for attention and loss. **(f)** Contiguous patch masking, in training and at inference, following TiRex [8]. The figure layout is inspired by Figure 1 of Chronos-2 [6].

Input preparation Values are normalized one variate at a time, then cut into patches of 32 steps. Targets $\mathbf{y}$ and past covariates $\mathbf{x}$ are standardized with *causal* running statistics. The mean μ_t and standard deviation σ_t at step t are computed from $y_{1:t}$ alone. An $\operatorname{arcsinh}$ transform follows, which compresses heavy tails while staying linear near zero. Chronos-2 popularized it for time series [6]. Known-future covariates $\mathbf{z}$ are observed over the whole span $1{:}T{+}H$ by definition, so they are standardized with statistics over that entire length.

In training, input and target are not scaled with the same statistics. For a forecast issued at T, dropping the variate index v, the model reads the left-hand side and predicts the right-hand side of

$$\tilde{y}_t = \operatorname{arcsinh}\left(\frac{y_t - \mu_t}{\sigma_t}\right) \quad \text{for } t \le T, \qquad \tilde{y}_{T+h} = \operatorname{arcsinh}\left(\frac{y_{T+h} - \mu_T}{\sigma_T}\right) \quad \text{for } h \in [H]. \tag{2}$$

Each input step is scaled with the statistics available at that step, its own value included, whereas every target step is scaled with the statistics frozen at T, which never include the target itself, since nothing later is available when the forecast is issued. The model must therefore predict any change of level or spread over the horizon itself. At inference, the prediction is brought back to the data space by inverting the same transform, $\hat{y}^q_{T+h} = \mu_T + \sigma_T \sinh\big(\hat{\tilde{y}}^q_{T+h}\big)$. No information past T reaches the model through scaling.

Missing values Time series often contain missing values. We track them with a binary mask alongside the values. Every non-observed step is excluded from the normalization statistics and the loss.

Embedding Each patch is embedded from three channels of 32 entries: the standardized values, a within-patch time index, and the validity mask of every step. The time index ranges linearly from 0 to $31/32$ over the patch. The three are concatenated and projected by a residual block. The output is added to a learned embedding of the role of the variate (either target $\mathbf{y}$, past covariate $\mathbf{x}$, or known-future covariate $\mathbf{z}$). The encoder is adapted from Chronos-2 [6]. No timestamp or frequency feature is required by the encoder.

Attention layers Covariate support hinges on the layer stack at the heart of t_0, which alternates time and variate attention. Both operate on the embedded patches rather than on the original time steps.

Time attention is causal self-attention along the patch sequence of each variate. It uses rotary position embeddings [66] indexed by patch position. There is one rotary unit per patch. The offset of the time step within a patch is only considered by the model through the embedding. The rotary embedding scales queries and keys by reciprocal position-dependent factors [68], so that each attention score decays exponentially with the distance between the two patches. Far-away patches thus weigh less, which keeps attention stable on contexts longer than any seen in training. The time attention is causal for target $\mathbf{y}$ and past covariates $\mathbf{x}$, while known-future $\mathbf{z}$ attend bidirectionally over $1{:}T{+}H$. It is how accessible information placed over the horizon is read.

Variate attention is self-attention across variates at each patch position, with no positional encoding to remain permutation equivariant since variates carry no natural order. The attention is asymmetric. While target $\mathbf{y}$ and $\mathbf{x}$ attend to one another, known-future $\mathbf{z}$ are read by the two other roles without attending back. This avoids a two-hop future information leakage that the alternating attention would otherwise open. Indeed, a known-future $\mathbf{z}$ attends over its whole span, so had it absorbed target content at some step s, a later time layer would carry that content to every earlier step $t < s$, and a later variate layer would hand it to the target at t.

Attention layers are arranged in a repeating pattern of two time-attention layers followed by one variate-attention layer. `t0-alpha` has 24 attention layers in total: 16 for time, 8 for variates. Each attention block uses pre-norm with RMSNorm [75] and is followed by a SwiGLU residual feed-forward layer [63]. They also normalize queries and keys per head before the attention product to prevent the softmax from saturating, following QK-norm [18, 29]. A final RMSNorm closes the stack.

Decoder The decoder turns the representation of a patch into a forecast. It is a residual block, a two-layer perceptron with a linear skip connection, applied to every patch in parallel. Under causal attention, the representation of the patch ending at step T summarizes everything observed up to T. The decoder reads it and predicts the next patch, the 32 steps $T{+}1, \ldots, T{+}32$, at the five native quantile levels $\mathcal{Q}_0 = \{0.1, 0.25, 0.5, 0.75, 0.9\}$. Every patch boundary therefore sits at information cutoff date. The output $\hat{\hat{y}}^{(v),q}_{T+h}$ is rescaled to the data space as described in Section 4.1.

Quantile crossing is a standard defect of independently estimated quantiles, usually repaired after the fact by rearrangement [14]. The decoder instead makes the five quantiles monotone by construction. It predicts the lowest level directly and every other level as a positive increment on the level below. For consecutive levels $q_k < q_{k+1}$ of $\mathcal{Q}_0$, $\hat{\hat{y}}^{\,q_{k+1}} = \hat{\hat{y}}^{\,q_k} + \mathrm{softplus}(r_{k+1})$, where r is the raw decoder output and $\mathrm{softplus}(u) = \log(1 + e^u)$ is strictly positive.

4.2 Training

Training data `t0-alpha` is pretrained on five corpora (Table 1), one real and four synthetic. While pretraining on synthetic data alone is an established alternative [21, 30, 46, 73], like most forecasting models, `t0-alpha` mixes synthetic and real data.

The real corpus is the GIFT-Eval pretraining split [2]. Its 89 datasets span domains like energy, cloud operations, transport, sales, climate, and healthcare. We rebalance the corpus. The ERA5 and CMIP6 climate collections were excluded for their size and homogeneity, and one cryptocurrency series for its extreme variability. Most of its series are univariate. Past covariates account for about a sixth of its values, and fewer than one series in ten carries more than one target.

The four synthetic corpora supply most of the multivariate series, each from one generator family. *Kernel-composition* generators sample Gaussian processes with randomly composed kernels, in the spirit of KernelSynth [5]. *Causal-graph* generators sample structural causal models in which nodes are time series and edges define causal dependencies among them. Nodes are then randomly assigned to targets, past covariates, or known-future covariates. This allows for the simulation of complex yet realistic data-generating processes, where each time series results from multiple interacting causes (see, e.g., [39, 55, 73] for more details). *Covariate-effect* generators build multivariate samples from hand-designed effect families with known ground truth: level shifts and closures, spikes, interactions between covariates, lag and lead responses, and decoy covariates carrying no signal, with both past and known-future covariates. A *mixing* augmentation combines real series into convex mixtures, in the spirit of TSMixup [5].

Table 1. Training corpora of `t0-alpha` and probability of each data pool over three different phases of the training. Series and target steps are computed as the total for each corpus.

				Draw probability		
Corpus	Covariates	Series	Target steps	0–40k	40–80k	80–150k
GIFT-Eval pretrain, rebalanced	past, in part	2.97M	24.9B	12%	10%	8%
Mixing	none	10.0M	21.2B	39%	35%	26%
Kernel composition	none	10.0M	81.9B	39%	35%	26%
Causal graph	past and future	1.00M	22.5B	9%	18%	36%
Covariate effects	past and future	0.10M	0.11B	1%	2%	4%

The mix of the five corpora changes over training, in the manner of curriculum learning [10]. Series are drawn from two pools. A multivariate pool is made of the causal-graph and covariate-effect corpora, and a pool of the other three, which is mostly univariate. A draw first picks a pool, then a series uniformly within it, so a corpus weighs in proportion to its series count within its pool. The multivariate pool receives 10% of the draws over the first 40k steps, 20% over the next 40k, and 40% from step 80k on (Table 1). The model is thus first exposed to mostly univariate series, and the share of multivariate ones grows from a tenth to two fifths. The schedule rests on the assumption that leveraging multivariate information is the harder skill, and easier to learn once the behavior of a single series is captured.

Each series undergoes a few transformations on the fly. A window of 8192 steps is cut at a random position, or the whole series when shorter. Its edges are jittered by up to one patch, so during training the model sees series padded at their start. Patch boundaries also fall on varying steps of the same time series whenever it is sampled multiple times. Missing values are flagged in the mask and their slot filled with zero.

Loss objective Since the model predicts quantiles directly, it is trained with the quantile loss that underlies the CRPS, as are Chronos-2 and the TiRex models [6, 8, 54]. Every patch boundary of a target variate is an information cutoff datetime T at which the decoder predicts the next 32 steps at the five native levels. Each predicted step is scored against its observation in the normalized space of Equation 2, both scaled with the statistics frozen at T. The CRPS of a quantile forecast is an integral of pinball losses over quantile levels [25, 38], which we discretize over $\mathcal{Q}_0$ by the centered rectangle rule. Dropping the variate index as in Equation 2, the loss of step h is

$$\ell_{T,h} = \sum_{q \in \mathcal{Q}_0} 2\, w_q\, \rho_q\Big(\tilde{y}_{T+h} - \hat{\tilde{y}}^{\,q}_{T+h}\Big), \qquad \rho_q(u) = u\,\big(q - \mathbf{1}\{u < 0\}\big), \tag{3}$$

where ρ_q is the pinball loss [36]. The weight w_q of a level is the width of the cell between the midpoints to its neighbors, the two outer cells extended to 0 and 1. $w = (0.175, 0.2, 0.25, 0.2, 0.175)$, summing to one. The pinball losses at 0.1 and 0.9 thus stand in for the whole tails, and nothing in training asks the model to place a level beyond them.

The model is trained for next-patch prediction. Every patch of a series, but the first, is considered as the target predicted from the context of the previous patches. The loss is computed on the time steps of the target patch. Each of the 32 steps of a predicted patch is scored against its observed value with Equation 3. Only target variates are scored. The model also predicts the past covariates, but their outputs are masked out of the loss, so a covariate influences a forecast through attention alone. Missing values and padded steps are discarded from the loss. The loss is then averaged per variate row of the mini-batch, then across rows. With $\mathcal{S}_r$ the scored positions (T, h) of row r and $\mathcal{R}$ the rows holding at least one,

$$\mathcal{L} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \frac{1}{|\mathcal{S}_r|} \sum_{(T,h) \in \mathcal{S}_r} \ell_{T,h}. \tag{4}$$

Packing As the length of time series varies across samples and domains, the training pipeline must cope with this variability. A simple strategy is to pad every series to a maximum length, which wastes compute on padding. Instead, following large language model training practice [19, 37, 60], time series are packed during training, as in Figure 3(e). Language model packing fills one axis, the sequence of tokens, because text is one-dimensional. t_0 packs in two dimensions. A time series is a 2D block whose dimensions are variate and time. A series with n variates and ℓ steps occupies a rectangle of n rows by ℓ columns. During training, time series are tiled in a pack of 128 variate rows by 8192 time steps. Several series land side by side along the time axis and several are stacked along the variate axis. Building a mini-batch this way is a two-dimensional bin-packing problem [42]. For `t0-alpha`, the packer draws a buffer of candidate series, fills the grid with those that fit the space left, and discards the remainder. Figure 3(e) shows one such grid. A multivariate series occupies a block of several rows, a univariate one a single row, and the cells no series fits are left unused.

The packing idea for time series originates from Toto, which stacks independent multi-target time series along the variate axis [15]. To avoid the spill of information from one time series to another, the packing adds a mask identifying the time series each time step of a mini-batch belongs to, similarly to Chronos-2 [6]. This mask is used both by the attention layers and the loss computation. Additionally, each time series is padded to a multiple of the patch size, so every patch includes information about a single time series. To our knowledge, t_0 is the first model family to rely on this two-dimensional time series packing for training.

Contiguous patch masking The training employs the contiguous patch masking strategy introduced by TiRex [8] and adopted since by TiRex-2, Toto-2, and TimesFM-3.0 [28, 35, 54]. Contiguous spans of target patches are blanked from the input, as in Figure 3(f), so the model has to reconstruct them from the surrounding context. Spans of one to sixteen patches are blanked at random on each target row independently. The training objective,

however, remains the next-patch prediction loss. It is computed at every patch boundary, whether or not the patches read by the model were blanked, which keeps a high proportion of patches contributing to the loss. Figure 3(f) shows one blanked span and the prediction every patch still emits for the next.

Training run `t0-alpha` was trained from scratch on a single node with four NVIDIA H100 80 GB GPUs, using data parallelism and mixed-precision `bfloat16`. The released checkpoint was saved at step 150k after 28 hours. Each GPU processes one pack of 128 variate rows by 8192 steps per optimization step. The model has been trained on 5.4×10^{11} observed time steps, counting every variate of every series, targets and covariates alike. The optimizer is AdamW [43] with a peak learning rate of 3×10^{-4}, a weight decay of 0.01. The learning rate follows a linear warmup over the first 5k steps, then a cosine decay down to 5×10^{-5} at step 150k. Gradient clipping was set at a global norm of 1.

4.3 Inference

At inference, the model must produce the forecast $\hat{y}_{T+h}^{(v),q}$ over a prediction length of H steps (Equation 1). Because it was trained with contiguous patch masking, two regimes are distinguished by this length. When H is at most 1024 steps, the horizon is appended to the context as blanked patches and a single forward pass yields the quantiles at every step $h \in [H]$ and every native level $q \in \mathcal{Q}_0$, the inference half of Figure 3(f). This already extrapolates the training regime, where at most 512 target steps were blanked at once. When H is larger, the model performs a rollout, following Toto-2 [35]. Forecasts are produced by chunks of 1024 steps, each chunk being folded back into the context, until the full horizon is covered.

In a rollout, the context must be extended with the previous forecast. Rather than feed a single point path back, we replicate the context into $|\mathcal{Q}_0| = 5$ paths, one per native level, and the path of level q is extended with its own quantile predictions $\hat{y}^{(v),q}$. So the uncertainty of one chunk is carried into the next. Each path then yields five quantiles. These values are pooled, weighted by the probability mass of their path and of their level, and reduced back to the five native levels as weighted quantiles. This reduced prediction is merged into the previous context and fed back to the model for the next iteration. The rollout thus follows several paths and reduces them at every chunk. This is the strategy of Chronos-2 [6], whereas Toto-2 feeds only the median back [35]. Reducing to the median is simpler, but it can oversmooth the forecast. Following one path per quantile level is expected to limit this smoothing, at the cost of a more expensive inference, since the batch is inflated by the number of paths.

The full stack has approximately 102M parameters. Table 2 summarizes the configuration of `t0-alpha`.

5 Evaluation

5.1 Public benchmarks

We evaluate `t0-alpha` on three public benchmarks: GIFT-Eval for univariate accuracy, fev-bench for accuracy with covariates, and TIME for accuracy on datasets assembled to lie outside public pretraining corpora.

The comparison is against other TSFMs. Every rank we report is to be understood within the list of released zero-shot checkpoints. The public leaderboards are wider. They include traditional statistical systems that are systematically outperformed, but also agentic systems, ensembles and fine-tuned entries. These systems generally route, ensemble or search over one or more pretrained forecasters rather than provide forecasts directly themselves. `t0-alpha` is a candidate component of such a pipeline rather than an alternative to one, so we leave those entries out of the ranking.

Table 2. `t0-alpha` architecture summary.

Parameters	≈102M
Layers	24 (16 time-attention, 8 variate-attention)
Layer pattern	2 time layers, then 1 variate layer
Embedding dim	512
Feed-forward dim	2048 (SwiGLU)
Attention heads	8
Patch size	32
Norm	pre-norm RMSNorm, QK-norm, final RMSNorm
Quantile head	monotone by construction (cumsum-of-softplus)
Positional encoding	RoPE by patch position (time axis only)
Native quantile levels	0.1, 0.25, 0.5, 0.75, 0.9
Single-pass horizon	up to 1024 steps
Trained blanked span	up to 512 steps

The results of this section mostly concern `t0-alpha`. We also report results on the public benchmarks for `t0-beta`, its successor. When no model is named, a result concerns `t0-alpha`. Among the released zero-shot models of the three tables, `t0-alpha` ranks 11th of 17 on GIFT-Eval by CRPS, 12th of 20 on fev-bench by skill score and 8th of 12 on TIME by CRPS. `t0-beta` would enter the same tables third, third and fifth, and it holds third place on GIFT-Eval under MASE as well as CRPS.

A rank alone hides how close these models are. TimesFM-3.0 leads GIFT-Eval at 0.4557 CRPS, and `t0-beta` passes Toto-2.0-2.5B by 0.0021 of CRPS, 0.45% in relative terms, and under MASE as well, with a tenth of its parameter count. `t0-beta` is 4.0% of CRPS short of the top of GIFT-Eval, 2.3 skill points short of the top of fev-bench and 3.5% of CRPS short of the top of TIME, and `t0-alpha` is 5.8% of CRPS short of second place on GIFT-Eval.

The model runs the same way on the three benchmarks. It reads a context of at most 8192 steps. Longer time series are trimmed to their last 8192 steps. Shorter ones are left-padded to the next multiple of the 32-step patch. The model returns the five native quantiles that are used for evaluation. Whenever a benchmark asks for another level, it is obtained by interpolation between the two neighboring native levels. Point metrics such as MASE are computed on the median.

One and only checkpoint serves the three benchmarks, as it does for most of the models we compare with. TiRex-2 is however an exception. It enters GIFT-Eval and fev-bench with a different checkpoint for each. It is the same architecture but trained on different data, so its rows in the two tables are less comparable than the others. Additionally, some models like Granite-PatchTST-FM or TiRex-2 do not appear in all the three benchmarks. When it is the case, we report model results only for the benchmark they have been publicly run on.

Several studies below run on a 12-pair subset of GIFT-Eval spanning all seven domains and frequencies from 15-minute to monthly, chosen so that peer checkpoints could be run under identical conditions within our compute budget. Table 13 in Appendix A lists the twelve datasets.

5.1.1 GIFT-Eval

GIFT-Eval [2] has become the standard benchmark for pretrained forecasting models. It comprises 97 forecasting tasks built from 23 datasets across seven domains, with frequencies from seconds to years and univariate as well

as multivariate series. A task pairs a dataset, sampled at one frequency, with a forecast term that sets the horizon length to predict. Each task provides a context from which to predict and a target to compare the forecast against.

We compute two metrics, CRPS and MASE, which measure the accuracy of the probabilistic forecast and of the median point forecast, respectively. MASE is the mean absolute error of the forecasted median over the target windows, scaled per series by the in-sample mean absolute error of the seasonal naive forecast. This scaling makes errors comparable across series of different magnitude. The CRPS is approximated using the weighted quantile loss and normalized by the target magnitude: for each of nine quantile levels, $q \in \{0.1, 0.2, \ldots, 0.9\}$, twice the pinball loss is summed over every series and horizon step of a dataset and divided by the summed absolute value of the target; the nine values are then averaged. Each metric is then normalized by the value the seasonal naive forecast obtains on the same dataset, frequency and horizon configuration. Consequently, an error of 1 indicates the method matches the baseline, and lower is otherwise better. The overall score is the geometric mean of these normalized values over the 97 configurations.

GIFT-Eval carries covariates on few of its series, but we use none. Every variate is forecast on its own, from its own past. We therefore use GIFT-Eval to measure univariate zero-shot accuracy. Consequently, the model could even get better results by using the covariates that are present. On this protocol `t0-alpha` reaches an aggregate CRPS of 0.4941 and a MASE of 0.7240. Table 3 places these scores next to those of the other zero-shot models, sorted by CRPS, under which `t0-alpha` is 11th of the 17 leaderboard models. Under MASE it would be 12th. `t0-beta` reaches a CRPS of 0.4738 and a MASE of 0.6865 on the same 97 pairs. It enters Table 3 third by CRPS and third by MASE, behind only TimesFM-3.0 and Granite-PatchTST-FM. It passes Toto-2.0-2.5B by 0.45% of CRPS, and under MASE as well, while carrying a tenth of its parameters. Against `t0-alpha` it lowers CRPS by 4.1% and MASE by 5.2%.

GIFT-Eval aggregates over tasks to produce a final benchmark value. However, investigating the individual tasks underlying this value can provide additional insights. We therefore split the benchmark by domain and frequency and compare `t0-alpha` with five models ranked above it. For each model, we compute the win rate of `t0-alpha`, defined as the fraction of the 97 dataset/term pairs in which its CRPS is strictly lower than that of the compared model, broken down by domain and frequency class (Figure 4). Over all pairs, `t0-alpha` performs on par with TiRex (win rate 0.51, CI 0.40–0.61) and TimesFM-2.5 (0.49, CI 0.39–0.60), and behind TiRex-2 (0.45, CI 0.35–0.56), Chronos-2 (0.43, CI 0.33–0.54) and Toto-2.0 (0.33, CI 0.24–0.42). Only the Toto-2.0 deficit is statistically unambiguous, its CI lying entirely below parity. By domain, `t0-alpha` beats both TiRex generations on web and cloud-operations telemetry and on nature data, and Toto-2.0 on sales. By frequency, it holds its own at hourly resolution, where it beats Chronos-2 and TiRex, but loses ground at sub-hourly resolution against Toto-2.0 and Chronos-2, and at daily, weekly and monthly frequencies against every model. However, it is not possible to draw more general conclusions for quarterly and yearly frequencies, which contain only one dataset/term pair each. Stratifying the ranking of Table 3 by horizon term adds one more fact: among its 17 leaderboard models, `t0-alpha` ranks 12th on the 55 short-term pairs, 9th on the 21 medium-term pairs and 6th on the 21 long-term pairs. The model climbs the table as horizons lengthen, consistent with single-pass decoding of long horizons (Section 4.3), which suggests that its long forecasts are more stable than those of its competitors.

5.1.2 fev-bench

fev-bench [64] complements GIFT-Eval for multivariate evaluation. Whereas GIFT-Eval scores every variate on its own, 68 of the 100 fev-bench tasks carry covariates or several targets. 42 tasks declare past or known-future covariates and 35 several targets, with some overlap between the two. Regarding data, fev-bench also extends GIFT-Eval. It reuses some datasets but also brings new time series to evaluate on. Each task is scored by the scaled quantile loss, which corresponds to the quantile loss over the same nine levels, divided per series by the in-sample mean absolute seasonal difference. It has the same denominator as the MASE for GIFT-Eval. The

Table 3. GIFT-Eval aggregated CRPS and MASE results (lower is better). Each row corresponds to a zero-shot model. Rows are sorted by CRPS. The best value of each column is in bold. `t0-alpha` is shaded green and `t0-beta`, its successor, blue.

Model	Params	CRPS ↓	MASE ↓
TimesFM-3.0	331M	**0.4557**	**0.6668**
Granite-PatchTST-FM	385M	0.4672	0.6846
`t0-beta`	256M	0.4738	0.6865
Toto-2.0-2.5B	2.45B	0.4759	0.6956
TiRex-2	82.5M	0.4781	0.6973
Toto-2.0-1B	1.04B	0.4784	0.6992
Tafsut	105M	0.4809	0.6926
Toto-2.0-313m	313M	0.4814	0.7028
Chronos-2	119M	0.4854	0.6978
TiRex	35M	0.4885	0.7158
TimesFM-2.5	231M	0.4903	0.7050
`t0-alpha`	102M	0.4941	0.7240
Toto-2.0-22m	21.9M	0.4963	0.7188
Moirai-2.0	11.4M	0.5164	0.7281
Toto-1.0	151M	0.5173	0.7501
TempoPFN	38M	0.5327	0.7875
TabPFN-TS	11M	0.5441	0.7709
Sundial	128M	0.5590	0.7502
Seasonal naive	—	1.0000	1.0000

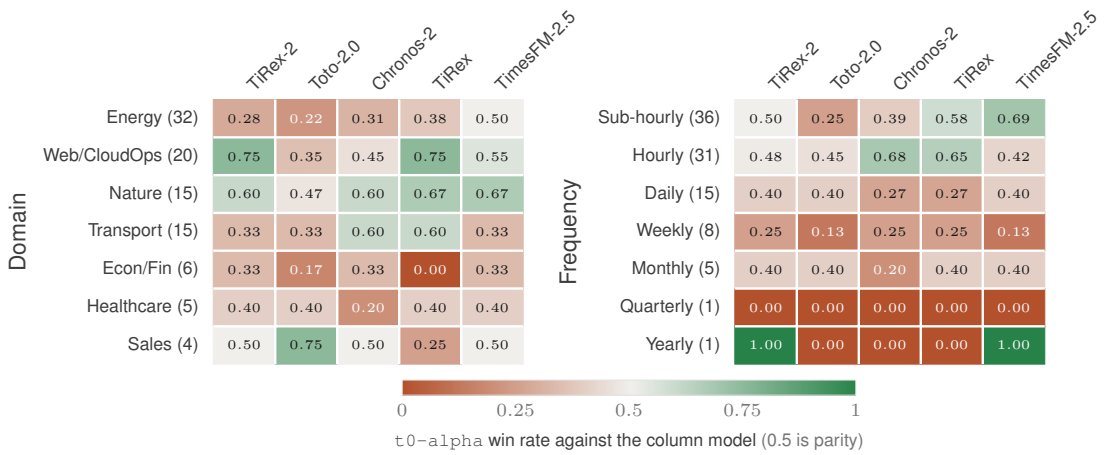

Figure 4. `t0-alpha` per-pair win rate against the five models ranked above it, by domain and by frequency class on GIFT-Eval. Green: `t0-alpha` wins the majority. Brown: it loses the majority. 0.5 is parity. Pair counts per group in parentheses.

benchmark reports two aggregates. The skill score of a model and the win rate. The former is one minus the geometric mean over tasks of the ratio between its loss and that of the seasonal naive baseline, each ratio clipped to $[0.01, 100]$ before aggregation. It is reported as a percentage, 0 matches the baseline and higher is better. The

latter is the share of task and opponent pairs in which the model attains a lower loss than the opponent, a tie counting one half. It depends on the models on the board and moves as entries are added, while the skill score remains fixed. Two imputation rules apply to every entry: a task a model fails is scored as seasonal naive, and a task whose dataset lies in the declared pretraining corpus of the model is scored as Chronos-Bolt. `t0-alpha` declares eight such tasks, drawn from the GEFCom load, M5, Favorita, KDD Cup 2022 and FRED-MD datasets, so its skill score under the rule of the benchmark, 42.2, is below the 42.9 its own forecasts obtain.

We use fev-bench to test how the model leverages covariates, following the design of the covariate ablation of Chronos-2 [6]. We run the same checkpoint twice on every task, once with the time series covariates passed and once with them dropped, so the difference isolates the covariate pathway. On the 58 tasks without covariates the two runs perform the exact same pass, which we verified exactly, so the effect rests on the 42 covariate-informed tasks. With covariates, `t0-alpha` attains a skill score of 42.2% and a win rate of 59.6% on all 100 tasks, and 39.7% and 62.7%, respectively, on the 42 covariate-informed tasks (Table 4). `t0-beta`, whose pretraining corpora were built to exclude every dataset declared by the benchmark, is zero-shot on all 100 tasks. Across all tasks, it attains 46.4% in skill score and 77.2% in win rate, and on the covariate-informed tasks, 46.1% and 83.5%, respectively. It ranks third by skill score on both slices, behind only TimesFM-3.0 and Chronos-2, and third by win rate on the covariate-informed tasks. The win rates count both of our models as entries on the leaderboard, so each is an opponent of the other: `t0-beta` beats `t0-alpha` on 81 of the 100 tasks. We refer to Figure 5 for a detailed breakdown of the win rate by opponent.

Table 5 shows that `t0-alpha` can translate operational context into substantial forecasting gains without retraining. Passing known-future covariates raises skill from 36.7 to 43.0 across 30 tasks, a gain of 6.3 percentage points, with improvements on 19 tasks. Passing past covariates raises skill from 32.9 to 35.6, a gain of 2.7 points, with improvements on 9 tasks. On German day-ahead electricity prices, supplying the grid operator's day-ahead forecasts of load and solar and wind generation reduces quantile loss by 51%. On daily Rossmann drugstore sales, supplying planned promotions, holidays and opening days reduces it by 43%. Both comparisons use the same `t0-alpha` checkpoint with and without covariates. The gains are not universal. Eleven of the 30 known-future tasks score worse with covariates, with the largest deterioration reaching 15% on a 15-minute electricity series with weather covariates. On M5, eight covariates covering product prices, calendar events, and food-stamp days change quantile loss by less than 2% in either direction at weekly and monthly granularities (Figure 6). The results demonstrate the value of conditioning on informative context while identifying selective use of noisy or uninformative covariates as a remaining challenge.

To measure further how sharply the model reads covariate alignment, we take the German price task and degrade its two forecast covariates, shifting them in time by up to ± 64 hours, permuting them, or dropping them (Figure 7). Permutation is performed by reordering the hours of a covariate. We draw a random permutation of the time index over the whole span the model reads, context and horizon alike, and apply it to that covariate values, independently for each of the two. The alignment with the price is destroyed, and so is the coupling between load and generation. The price and its history are untouched. Loss degrades smoothly with misalignment, roughly doubling from perfect alignment to a 16-hour shift, and even permuted covariates beat dropping them, since their marginal distribution still carries level information. The model therefore keys on the point-by-point alignment with the target, at single-step resolution, rather than on their presence.

Table 4. fev-bench skill score and win rate, in percent, on all 100 tasks and on the 42 covariate-informed tasks (higher is better). Win rates are computed over the models of this table, some of which the official leaderboard does not carry, so they can differ from the values reported there. Each row corresponds to a zero-shot model. Rows are sorted by skill score on all tasks. The best value of each column is in bold. `t0-alpha` is shaded green and `t0-beta`, its successor, blue.

	All tasks (100)		Covariate-informed (42)	
Model	**Skill (%) ↑**	**Win rate (%) ↑**	**Skill (%) ↑**	**Win rate (%) ↑**
TimesFM-3.0	**48.7**	**87.0**	**48.2**	**88.2**
Chronos-2	47.3	81.3	47.0	85.0
`t0-beta`	46.4	77.2	46.1	83.5
TiRex-2	45.5	77.8	44.8	81.0
Toto-2.0-1B	44.4	76.7	38.9	68.3
Toto-2.0-2.5B	44.4	77.5	38.8	69.0
Toto-2.0-313m	44.2	75.1	38.7	66.7
TabPFN-TS-3	43.1	61.8	43.3	67.4
TS-ICL	43.1	62.4	41.9	65.6
Toto-2.0-22m	42.9	68.2	37.6	61.2
TiRex	42.6	70.0	38.7	68.5
TimesFM-2.5	42.2	66.1	37.4	62.6
`t0-alpha`	42.2	59.6	39.7	62.7
TabPFN-TS	41.5	56.1	42.5	65.9
CITRAS-FM	41.2	59.5	39.0	64.1
Toto-1.0	40.8	57.3	35.2	51.1
Toto-2.0-4m	40.7	56.5	35.4	50.7
FlowState	39.9	59.3	36.7	61.1
Moirai-2.0	39.3	51.6	36.6	54.4
Chronos-Bolt	38.9	51.1	35.9	52.5
Sundial	33.4	33.6	28.0	34.4
Seasonal naive	0.0	12.9	0.0	12.8

Table 5. Covariate lift on fev-bench: skill of the same checkpoint with covariates dropped and with covariates passed, by covariate subset. Confidence intervals are 95% bootstrap intervals over tasks.

Subset	Tasks	Skill without	Skill with	Lift (95% CI)	Tasks improved
Known-future covariates	30	36.7	43.0	$+6.3\ [+2.2, +10.9]$	19 of 30
Past covariates only	12	32.9	35.6	$+2.7\ [+1.0, +4.2]$	9 of 12

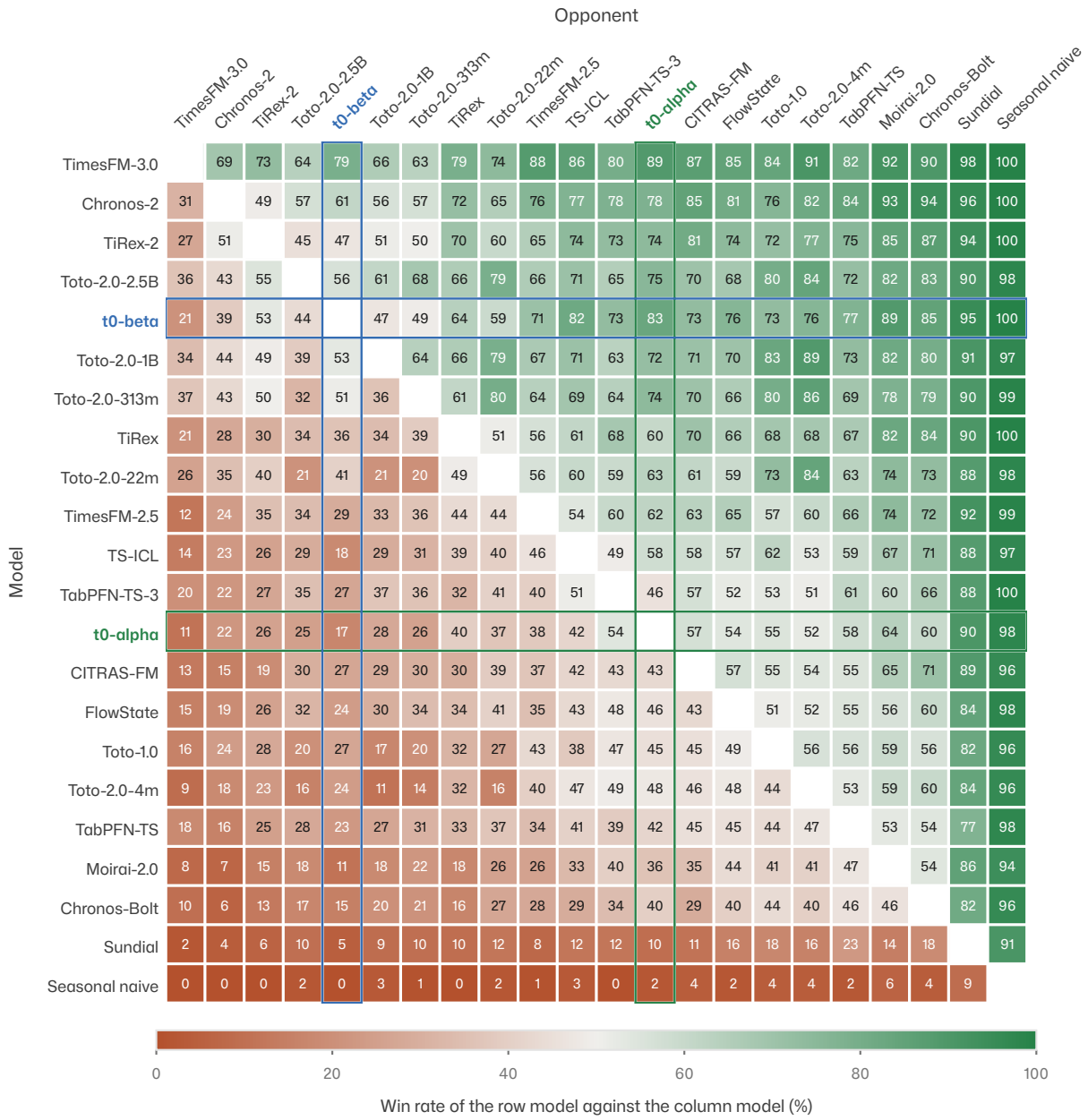

Figure 5. Pairwise win rates on fev-bench, all 100 tasks. It represents the share of tasks on which the row model attains a lower scaled quantile loss than the column model, a tie counting one half. The `t0-alpha` row and column are outlined in green, those of `t0-beta` in blue.

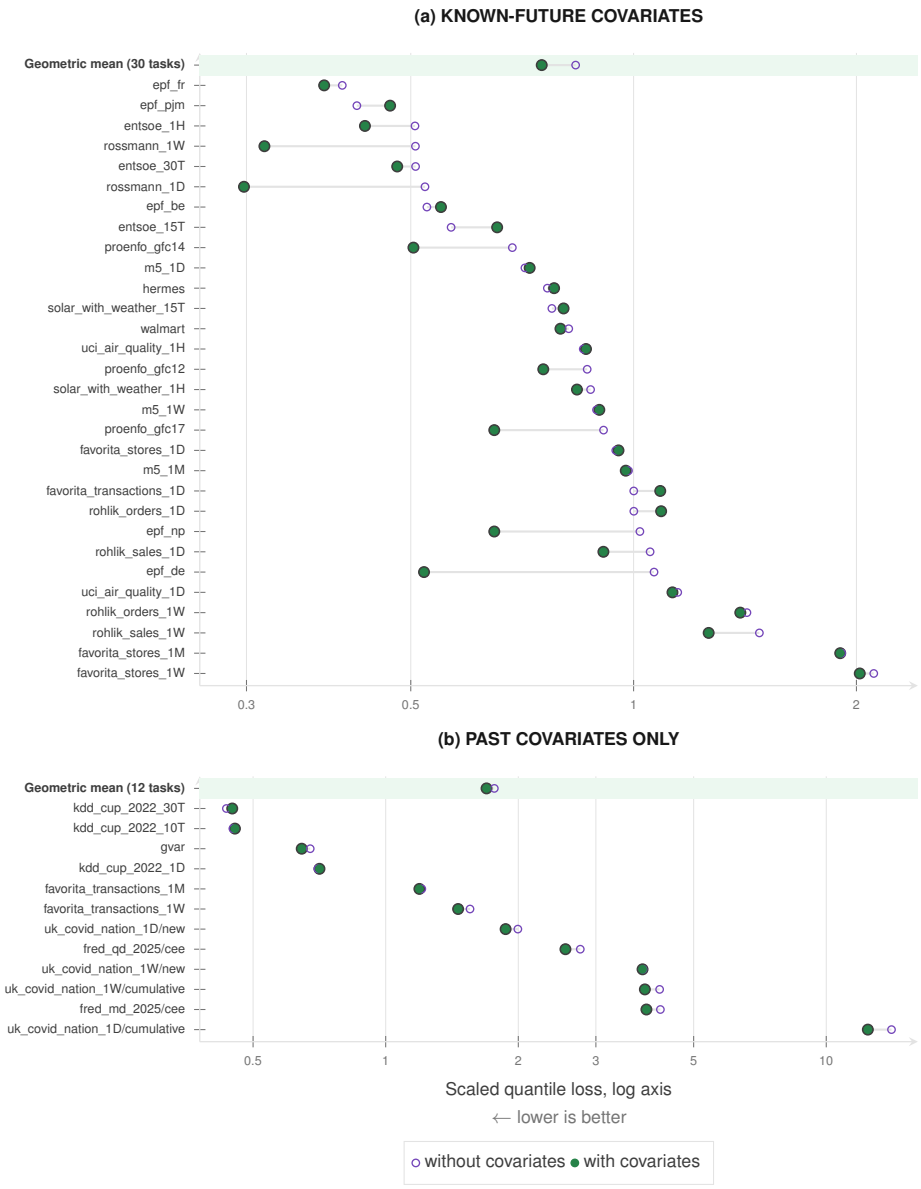

Figure 6. Covariate lift per fev-bench task. Scaled quantile loss of the same checkpoint without covariates (hollow) and with them (green), on a log axis so that the length of the line joining them is the relative change. Within each covariate subset, tasks are sorted by their loss without covariates, the lowest at the top, so the hollow marks read as a difficulty ordering. The shaded row is the geometric mean over the tasks of the subset.

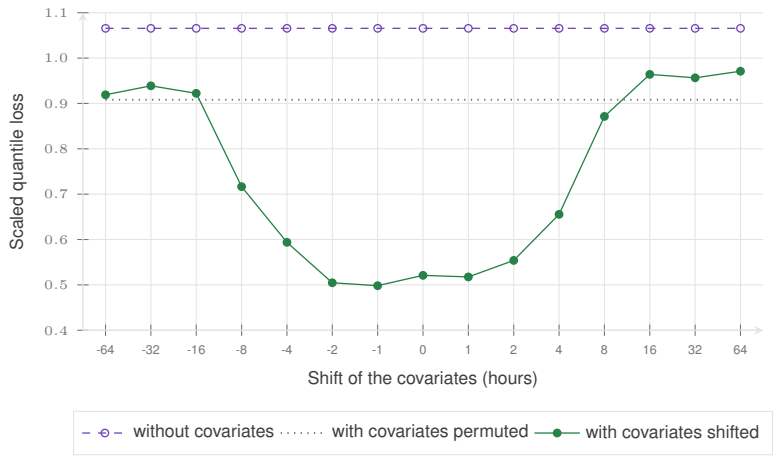

Figure 7. Alignment sweep on the German day-ahead electricity-price task: scaled quantile loss as the two forecast covariates, load and solar-plus-wind generation, are shifted in time by the lag on the x axis (green), randomly permuted in time (dotted), or dropped (hollow). Zero is the published alignment.

5.1.3 TIME

TIME [58] keeps the scoring of GIFT-Eval but replaces its data. MASE and CRPS are computed and normalized by seasonal naive as on GIFT-Eval, then aggregated by geometric mean over 98 tasks. A task pairs one of 50 datasets, each sampled at one frequency, with a horizon set per frequency and application. The 50 datasets are new and independent from GIFT-Eval. They span over the period between 2020 and 2025. There are no time series with covariates in the benchmark, but 74 of the 98 tasks involve multiple targets.

On the 98 tasks `t0-alpha` attains a normalized MASE of 0.685 and a CRPS of 0.572 (Table 6). `t0-beta` attains a MASE of 0.665 and a CRPS of 0.555, placing it 5th in Table 6 by CRPS and 6th by MASE, 3.5% of CRPS short of the lead.

Table 6. TIME aggregated CRPS and MASE, normalized by seasonal naive per task and combined by geometric mean over the 98 tasks (lower is better). Rows are the zero-shot models that Tables 3 and 4 also report, sorted by CRPS. The best value of each column is in bold. `t0-alpha` is shaded green and `t0-beta`, its successor, blue.

Model	CRPS ↓	MASE ↓
TimesFM-3.0	**0.5363**	**0.6398**
Toto-2.0-2.5B	0.5394	0.6419
Toto-2.0-313m	0.5423	0.6444
Toto-2.0-1B	0.5446	0.6448
`t0-beta`	0.5551	0.6647
Chronos-2	0.5563	0.6620
Toto-2.0-22m	0.5616	0.6703
TimesFM-2.5	0.5674	0.6686
`t0-alpha`	0.5721	0.6848
TiRex	0.5731	0.6831
Toto-1.0	0.5839	0.6980
Moirai-2.0	0.5885	0.7030
Sundial	0.6634	0.7575
Seasonal naive	1.0000	1.0000

TIME also scores models on groups of variates that share a temporal pattern, among the following: trend strength, trend linearity, seasonal strength, seasonal correlation, residual autocorrelation, spectral entropy and stationarity. Each continuous feature is cut at its median over the benchmark variates, and the score of a model on either side of the cut is the geometric mean of its normalized MASE over those variates.

Like every model in Figure 8, `t0-alpha` does better on strongly trended and strongly seasonal variates, and stationarity does not move its score.

5.2 Model capability analysis

5.2.1 Calibration

The public benchmarks above score accuracy. This section tests whether the model quantiles are calibrated. A quantile forecast at level q is calibrated if the realized value falls below it a fraction q of the time [26]. Calibration is what lets a forecast be used as a probability across decisions. For instance, a buyer who covers the 0.9 quantile

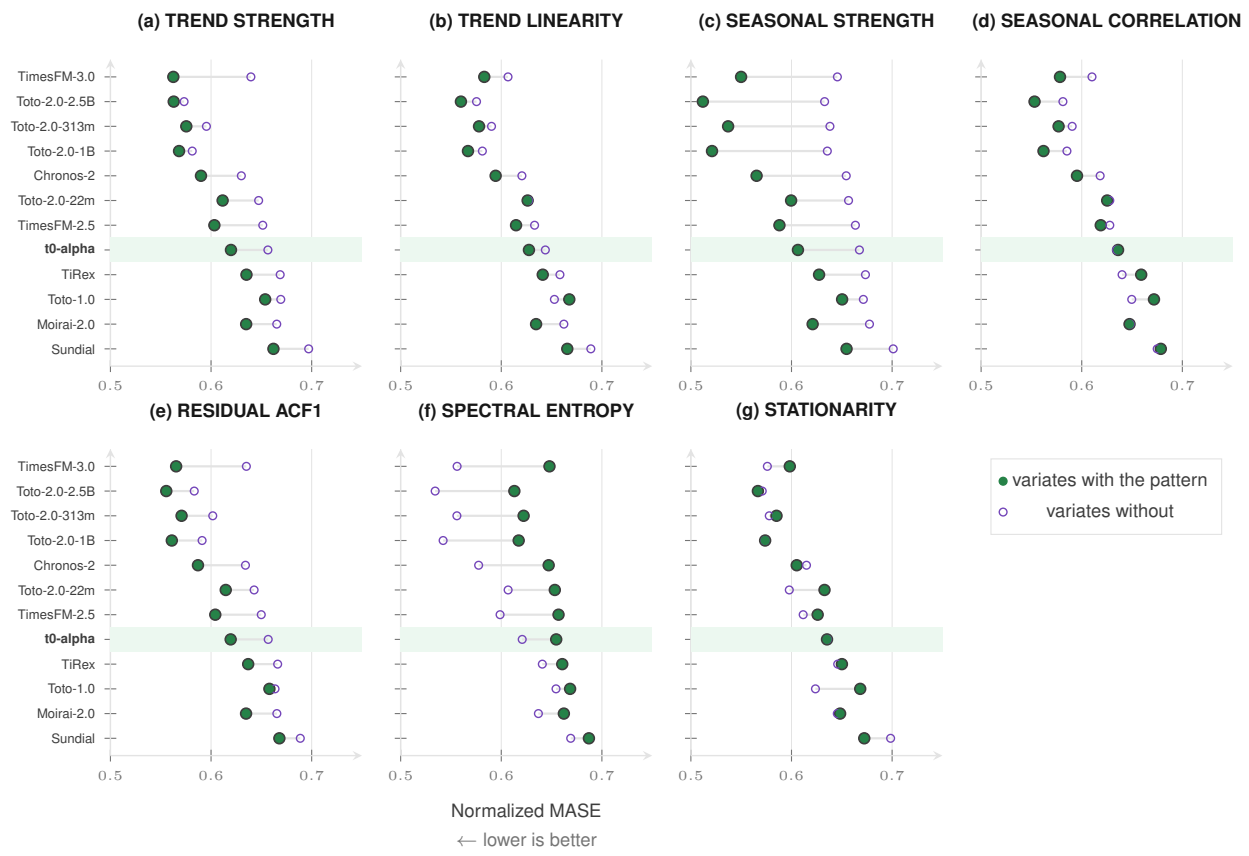

Figure 8. TIME pattern-level results for the models of Table 6, in the row order of that table: their overall normalized CRPS on the full benchmark, most accurate at the top. The x axis, shared by every panel, is the MASE normalized by seasonal naive on the variates that carry the pattern (filled) and on those that do not (hollow). Lower is better. **(a)–(f)** cut a continuous feature at its median over the variates of the benchmark; **(g)** is the stationarity flag of an augmented Dickey-Fuller (ADF) test.

of demand expects to run short one time in ten, whatever the series. Calibration is not sufficient on its own. A model that ignores the context and emits the unconditional distribution of the series is calibrated in this sense but carries no informative forecast on the horizon. Eventually calibration has to be read alongside accuracy, the sharpness of the distribution [26]. We measure calibration through coverage, the empirical fraction of outcomes that fall below a quantile or inside the band between two quantiles. A band can reach its nominal coverage while both its quantiles are off in the same direction, so we report coverage per level as well as per band.

To assess calibration, we generate forecasts over all 97 dataset/term tasks without using any covariates. For every forecast step we record whether the observation falls below each of eleven quantile levels: 0.1, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75, 0.8 and 0.9. It corresponds to the five native levels the model produces plus the six the package interpolates between them. We also check whether it falls inside the nominal 10–90 and 25–75 bands. We aggregate the results in two ways. First, we pool over all forecast steps, which weights a pair by the number of steps it contributes. Second, we aggregate as an unweighted mean over datasets, which gives a small dataset the same weight as a large one. Figure 9 shows the coverage of every pair and its mean.

Empirical coverage sits below nominal for both bands (Figure 9). Pooled over all forecast steps, the 10–90 band covers 0.784 of outcomes against a nominal 0.80 and the 25–75 band 0.478 against 0.50. The intervals are therefore too narrow by 0.016 and 0.022. Averaged over datasets they cover 0.777 and 0.482. The pooled median is exact, outcomes fall below it 0.500 of the time. Per level, pooled coverage stays within 0.014 of nominal

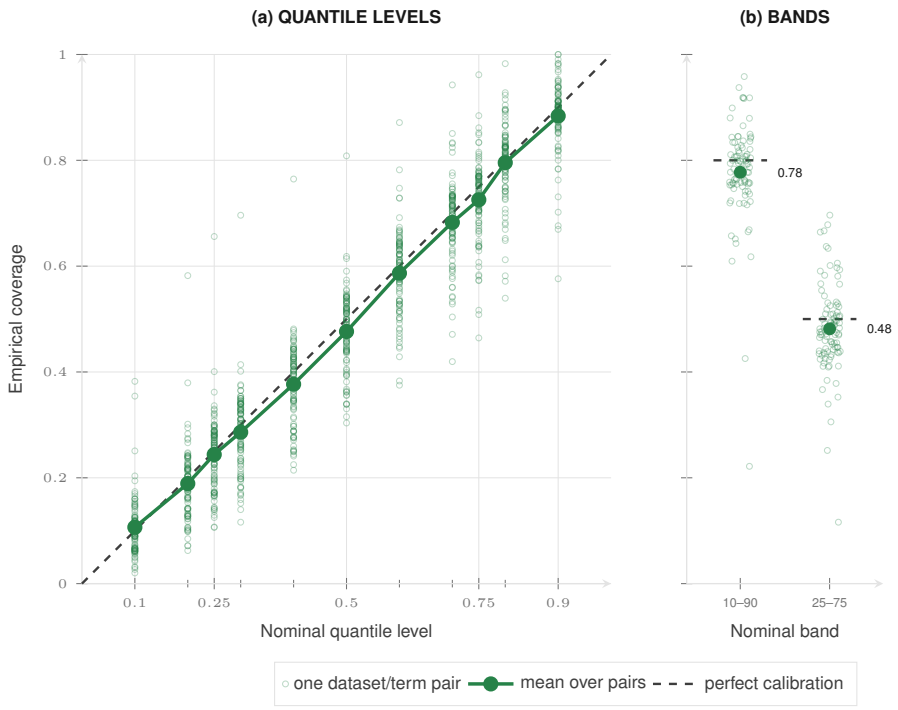

Figure 9. Calibration of `t0-alpha` on GIFT-Eval. Nominal against empirical coverage (a) at the eleven quantile levels and (b) for the two nominal bands. Faint hollow marks are single dataset/term pairs, filled marks the mean over pairs, dashed lines perfect calibration. The labeled levels 0.1, 0.25, 0.5, 0.75 and 0.9 are native. The six between them are interpolated.

at all eleven levels. The six interpolated levels are no worse than the native ones. Their largest pooled deviation is 0.011, the largest native one 0.013, so interpolation does not degrade coverage. Averaged over datasets, every level from 0.2 upwards sits 0.004 to 0.025 below nominal, while pooling brings them back to nominal. The datasets with few forecast steps, which pooling down-weights, are the ones whose quantiles run low.

5.2.2 Impact of quantile extrapolation on calibration

The model emits five quantile levels, and the released package turns them into any level a user requests. A requested level that matches a native one is returned as is. A level between two native ones is linearly interpolated between them. A level outside $[0.1, 0.9]$ is clamped to the nearest native level, so every tail request inherits the coverage of the 0.1 or the 0.9 quantile. Pooled over GIFT-Eval, a requested 0.01 quantile covers 0.109 of outcomes and a requested 0.99 quantile 0.893 (Table 7), an order of magnitude from nominal on the left tail.

We test an alternative that needs no training: the exponential tails of the incremental quantile function (IQF) of Park et al. [52], applied post hoc to the five native levels. Interior levels keep the released interpolation bitwise, so every other number in this report is unaffected, and each tail is pinned through the two outermost native levels on its side, where the original method learns a tail parameter. Figure 10 compares the two strategies over the 97 pairs. With IQF tails the 0.05 and 0.95 levels become close to calibrated, the coverage error at 0.01 drops sixteen-fold, and no extrapolated quantile crossed a native level across the benchmark. It improves accuracy. Normalized quantile loss at the extreme levels roughly halves, 0.064 against 0.110 at 0.01 and 0.088 against 0.164 at 0.99. Residual gaps remain, 0.016 coverage at the 0.01 level and 0.984 at 0.99, and the native levels are untouched by construction. Pinning the tails through two fixed levels is the simplest form of the method, and a learned tail parameter would likely close most of the remaining gap.

Table 7. Empirical coverage of requested tail levels on GIFT-Eval (97 pairs, pooled over forecast points), under the released flat-clamp extrapolation and post-hoc IQF exponential tails. All interior levels are identical between mechanisms.

Nominal level	0.01	0.02	0.05	0.95	0.98	0.99
Flat clamp (released)	0.109	0.109	0.109	0.893	0.893	0.893
IQF tails	0.016	0.025	0.054	0.948	0.975	0.984

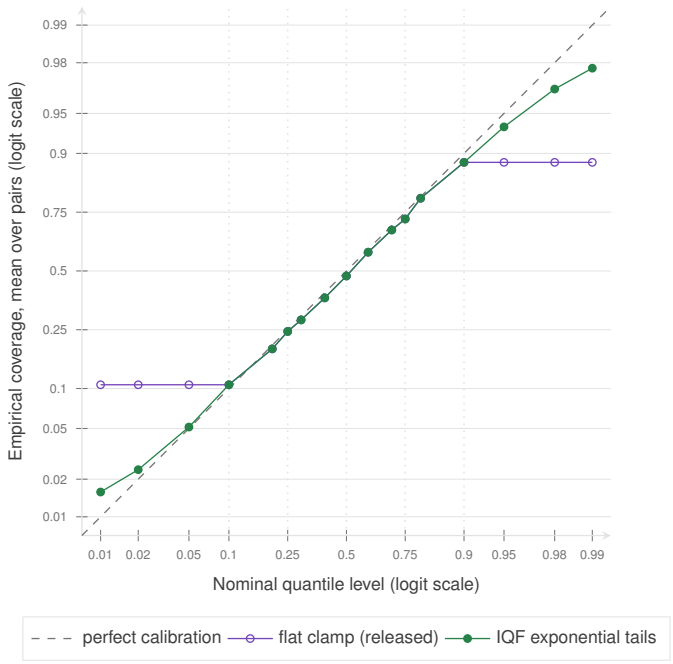

Figure 10. Reliability of `t0-alpha` on GIFT-Eval. Nominal against empirical coverage, averaged over pairs, at the 17 query levels from 0.01 to 0.99 under the released flat clamp (hollow) and the post-hoc IQF tails (filled). Both axes are on a logit scale. The dashed diagonal is perfect calibration and the dotted verticals mark the five native levels.

How a model answers a tail request is an architectural choice, and it splits the field. Chronos-2 trains its head on 21 levels including 0.01 and 0.99, so extreme levels are learned outputs [6]. Toto draws 256 samples from a parametric output distribution and reads empirical quantiles off them [15]. TiRex emits the nine deciles and its package clamps beyond $[0.1, 0.9]$ [8], as ours does. We run the three public checkpoints through the coverage protocol on the 12-pair subset (Table 13), each reproducing its published CRPS, and restrict `t0-alpha` to the same pairs (Table 8, Figure 11), so these numbers differ slightly from the 97-pair ones above. The models that can move past their native levels, Chronos-2, Toto and `t0-alpha` with IQF tails, follow the diagonal, while TiRex and the released package sit on flat segments at 0.099 and 0.893. On loss, `t0-alpha` with IQF tails is the most accurate at the 0.01 level, 0.048 against 0.071 to 0.072 for the three competitors, and ties Toto at 0.99, 0.114 against 0.112. Toto buys its tails with roughly two hours of sampling on this subset, where the IQF tails are a closed form over five numbers the head already emits. The interior tells the reverse story. The 10–90 band of `t0-alpha` is the closest to its nominal 0.80 of the four models, 0.798 against 0.794 for TiRex and 0.783 for Toto, while the intervals of Chronos-2 run narrow, 0.744, and its 0.1 level over-covers at 0.158: well-placed extremes on a less calibrated interior.

Table 8. Tail coverage and interior calibration across public checkpoints on the 12-pair subset (mean over pairs). Tail columns give empirical coverage at the requested level; band columns give exact interval coverage against nominal 0.80 and 0.50.

Model	0.01	0.05	0.95	0.99	10–90	25–75
`t0-alpha` (clamp)	0.091	0.091	0.889	0.889	0.798	0.513
`t0-alpha` (IQF)	0.008	0.038	0.937	0.976	0.798	0.513
Chronos-2	0.009	0.072	0.947	0.984	0.744	0.439
TiRex	0.099	0.099	0.893	0.893	0.794	0.509
Toto	0.013	0.048	0.930	0.976	0.783	0.467

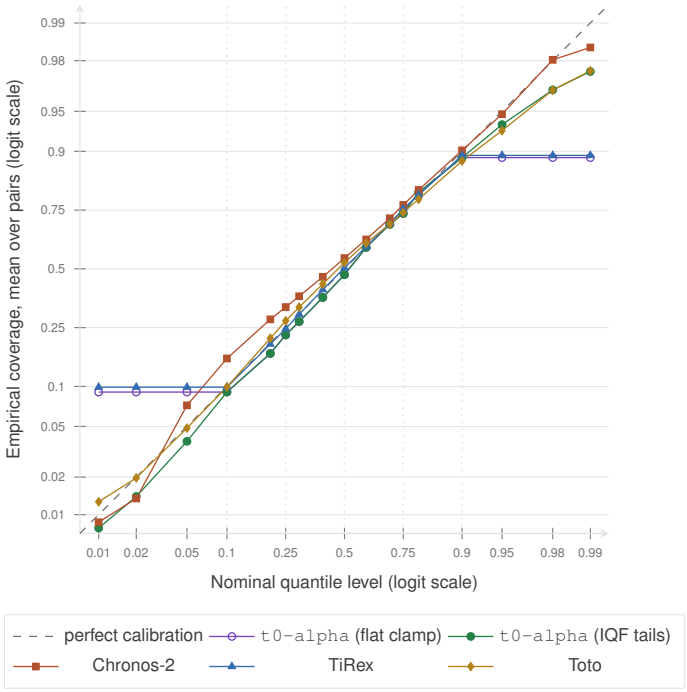

Figure 11. Reliability across public checkpoints on the 12-pair subset at the same 17 query levels, mean over pairs, on logit axes. The dashed diagonal is perfect calibration and the dotted verticals mark the native levels of `t0-alpha`. TiRex and the released `t0-alpha` package share the flat clamp and sit on horizontal segments beyond 0.1 and 0.9; Chronos-2, Toto and `t0-alpha` with IQF tails follow the diagonal.

5.3 Robustness to missing data

Operational history has gaps, and many foundation models, t_0 among them, accept them by design: missing values are treated as unobserved rather than imputed (Section 4.1). We quantify what they cost at inference. On the 12-pair subset (Table 13) we inject missingness into the context window at rates from 10% to 75% under two patterns, randomly scattered points and contiguous blocks that mimic sensor outages, both masking the same number of points at a given rate, and measure CRPS relative to the unmodified context of the same pair. Chronos-2 and TiRex run through the same protocol with identical seeded masks, each scored against its own clean baseline, and both reproduce their published per-pair CRPS before injection, Chronos-2 exactly and TiRex within 2% on 9 of 12 pairs. Table 9 and Figure 12 give the result. Deleting half the history raises the CRPS of `t0-alpha` by 1.31× for scattered points and 1.28× for blocks, and deleting three quarters by 1.56× and 1.42×.

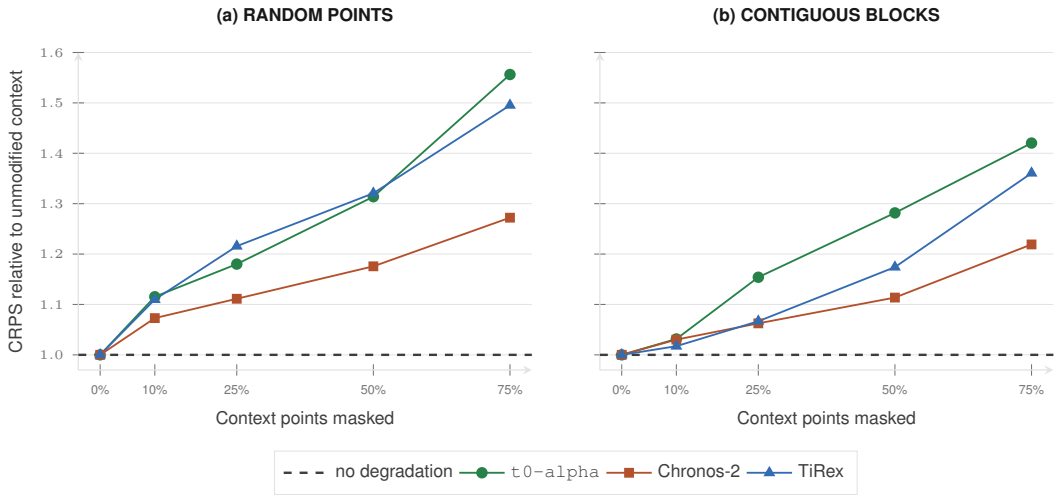

Figure 12. CRPS relative to the unmodified context of each model against the share of context points masked, geometric mean over the 12-pair subset, for (a) randomly scattered points and (b) contiguous blocks. Every model sees identical injected masks. The dashed line is no degradation.

Chronos-2 degrades least at every rate and pattern, 1.27× at 75% scattered where `t0-alpha` loses 1.56× and TiRex 1.50×. Part of its margin comes from its submission protocol, which forecasts the series of a dataset jointly, so signal surviving in one series compensates masks in another. Closing this gap is ongoing work. Against our expectation, scattered points cost more than blocks for all three models at every rate: scattered missingness touches every patch, while blocks leave most patches fully observed. The hard cases are shared. covid_deaths at 75% scattered costs `t0-alpha` 6.5×, TiRex 5.7× and Chronos-2 4.4×, since too little history survives to anchor a forecast, while high-frequency operational series barely move for any model.

Table 9. Relative CRPS under context missingness (geometric mean over the 12-pair subset; 1.00 = the unmodified context of each model). Chronos-2 and TiRex run under identical injected masks.

Pattern	Model	10%	25%	50%	75%
Random points	`t0-alpha`	1.12	1.18	1.31	1.56
	Chronos-2	1.07	1.11	1.18	1.27
	TiRex	1.11	1.22	1.32	1.50
Contiguous blocks	`t0-alpha`	1.03	1.15	1.28	1.42
	Chronos-2	1.03	1.06	1.11	1.22
	TiRex	1.02	1.07	1.17	1.36

5.4 Context efficiency

Missing data removes points from a fixed window. A complementary question is how short the window can be before accuracy suffers. Published context ablations start at 1024 points [31]. However, many operational regimes may have even shorter context. We thus measure how the model reacts to short contexts. On the 12-pair subset (Table 13), we cap the context of every series at its $c \in \{64, 128, \ldots, 8192\}$ most recent points and score CRPS relative to the same pair at the 8192 cap, which is the training window length. Because most series are shorter than the larger caps, we report the realized median context alongside the nominal cap, and a

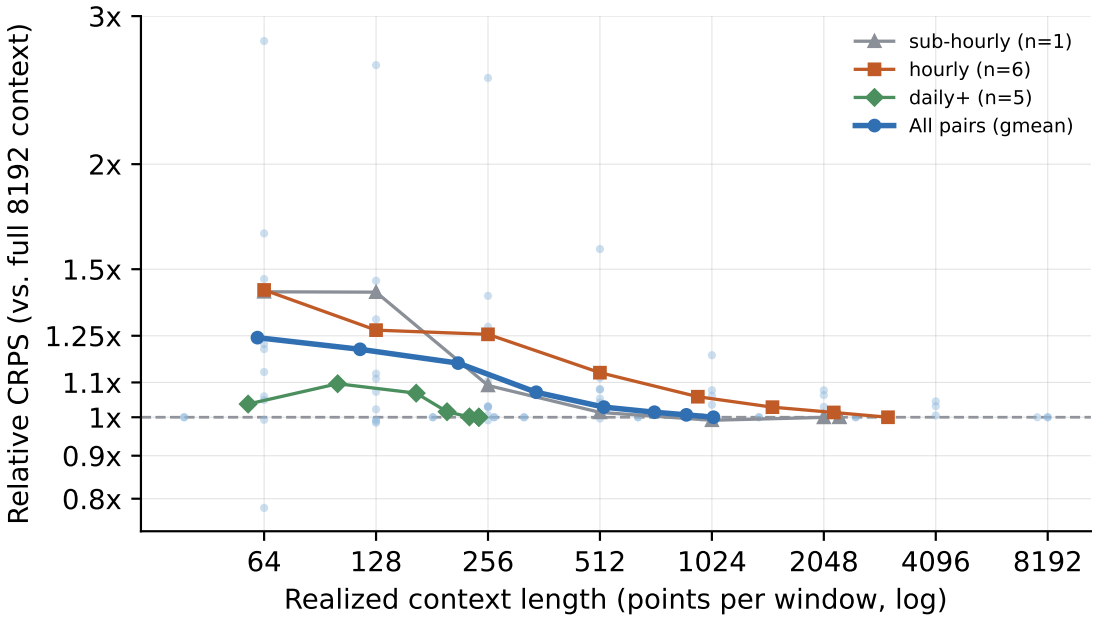

Figure 13. CRPS relative to the full 8192-point context cap vs. realized context length on the 12-pair subset (log x-axis). Lines are geometric means overall and per frequency band. Points are dataset/term pairs.

cap where the series of a pair are already all shorter inherits the score of the previous cap. The geometric-mean realized context at the full cap is roughly 1,000 points.

Figure 13 shows the curve. The model is within 5% of full-context accuracy from a 1024-point cap and within 1% from 4096. The cost of short context concentrates in high-frequency series: at 64 points, hourly pairs lose 42% while daily-and-slower pairs lose 3.7%, because 64 daily points hold several seasonal cycles that 64 hourly points do not. Truncation is not uniformly harmful. covid_deaths scores 22% better from its last 64 points than from its full 182-point history, where early-pandemic dynamics mislead more than they inform, and three operational pairs improve by 1–2% when truncated. These results show the model does not necessarily need long histories to be competitive at daily and slower frequencies.

5.5 Rollout strategy

At inference, the model must forecast over a horizon of H steps. Because it is trained with contiguous patch masking (CPM), it can predict several patches ahead in one pass. Following Toto-2 [35], the model falls back on a rollout whenever H is too long for one pass. In practice `t0-alpha` predicts 1024 steps, i.e., 32 patches, in one pass and it rolls out autoregressively beyond that. During the rollout it keeps one path per quantile level instead of reducing to the median between iterations (Section 4.3).

In what follows we compare different rollout strategies on a subset of the GIFT-Eval corpus. Specifically, we consider the sub-hourly datasets whose series are long enough: LOOP Seattle and Bizitobs-L2C at 5 minutes, solar and Jena weather at 10 minutes, electricity, ETT1, and ETT2 at 15 minutes, with at most 32 series per dataset, 138 in total. We request a forecast of 3072 steps given a context of 8192 steps. To follow accuracy along the horizon, we cut it into twelve windows of 256 steps and compute, for each dataset and each window, the CRPS as GIFT-Eval does. We compare four rollout strategies: the released 1024-step single pass re-anchored on the 8192 true steps before each iteration, which we use as baseline; the median-path reduction during the rollout; the quantile-path reduction used for inference in `t0-alpha`; and a single pass with the CPM mask extended to

the full horizon, three times the released limit. For reference, we also report an oracle forecast one patch ahead, which predicts each 32-step patch from the 8192 true steps before it. Curves are geometric means over the seven datasets. The first 1024 steps are the same forward pass under the baseline and the three strategies, so their ratio is one by construction.

The baseline and the one-patch-ahead forecast stand apart from the other three. The baseline re-anchors on the 8192 true steps preceding each of its three iterations, and the one-patch-ahead forecast on the 8192 true steps preceding each 32-step patch. These two refresh their context with observations that lie inside the horizon, so neither is feasible for a forecast made at a single origin. They show how the model would perform with access to that future information and bound the three strategies that do not use it: the median path, the quantile paths and the extended single pass, which answer the 3072-step request from the context available at the origin.

Figure 14 shows the three strategies relative to the baseline. Up-to-date context makes the clearest difference. The optimal forecast, one patch ahead from the true context, scores 0.54 times the baseline over the horizon. The baseline itself re-anchors on the truth only every 1024 steps, and the rollouts, which never do, score 2.5 to 2.6 times the optimal over the rolled-out steps. The cost of forecasting without fresh context is large. Averaged over datasets, the rollouts score 1.4 to 1.5 times the baseline on the third iteration and 1.6 to 1.8 times in their worst window, steps 2048–2304. The rollout with quantile paths is almost systematically below the median path. In seven of the eight rolled-out windows and on six of the seven datasets, by 4.8% on average. It is marginally better than the one pass over the full horizon, 1.31 against 1.32, ahead in five of the eight windows and on four of the seven datasets. Both the model and the rollout are stable. The one pass decodes three times the released horizon without breaking down. No rollout strategy drifts away from the others as the iterations proceed.

The lever on long horizons is therefore context, not the reducer: the strategies differ by a few percent, re-anchoring on observed data every 1024 steps removes the 30% the rollout costs, and re-anchoring every patch removes a further factor of 1.9. That lever belongs to the user who can re-issue the forecast as observations land. Whoever has to commit to the whole 3072-step horizon at one origin pays the rollout cost. Between the strategies, the median path is dominated, less accurate than the one pass at 2.4 times its inference time. The quantile paths cost 14 times the one pass and buy a 3% gain on the third iteration only, so the one pass is the cheap mode up to three times the released horizon and the quantile-path rollout the accurate one at the far end of the horizon.

5.6 Use cases

5.6.1 Hourly electricity demand in Australia

This benchmark comes from skforecast [3], a Python forecasting library with a scikit-learn interface. Its user guide on foundation models [4] backtests eight zero-shot models on the half-hourly electricity demand of Victoria, Australia. The data are taken from the `tsibbledata` collection [49] and aggregated to hourly means. The goal is to predict electricity demand given the temperature, and a holiday indicator. Temperature and holiday are supplied over the forecast horizon as known-future covariates, so a model receives the realized temperature of the day it forecasts. The guide backtests over one-day windows from October 31 to December 30, 2014, 61 windows in total. Each window forecasts the next 24 hours from the preceding 500 observations, three weeks of hourly history. The score is the mean absolute error, in megawatts, of the forecast of each model over all windows. For most models the forecast corresponds to the median. TimesFM, TabICL, and Nori however directly produce a mean prediction. Whenever possible, we reproduced the results of the guide with its own code, on the same backtest. Table 10 shows the results.

The guide rightly warns about leakage. The GIFT-Eval pretraining corpus [2], on which `t0-alpha` and many other public models train, holds the Victoria demand series twice. The `australian_electricity_demand`

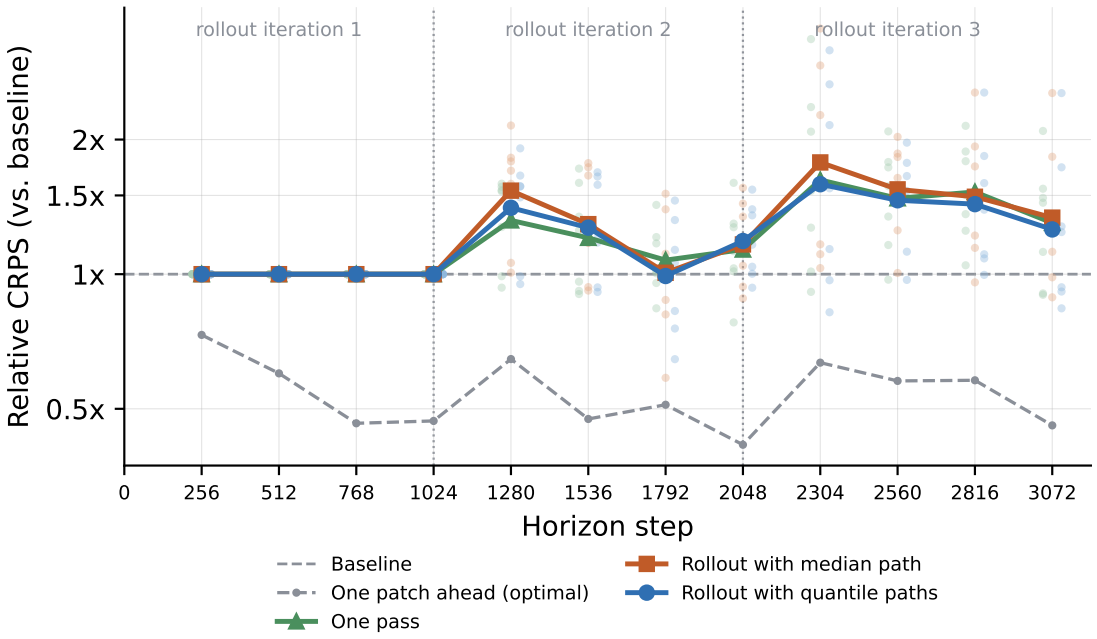

Figure 14. Rollout strategy on long horizons. CRPS per 256-step window of the horizon, relative to the baseline, on 138 series from the seven sub-hourly GIFT-Eval datasets long enough for an 8192-step context plus a 3072-step horizon. *Baseline*: the model predicts 1024 steps ahead in a single pass from the most recent context. *One patch ahead (optimal)*: the model predicts a single patch, 32 steps ahead from the most recent context. *One pass*: the model predicts all the extended horizon with CPM mask covering the full 3072 steps. *Rollout with median path*: the median folded back at each iteration. *Rollout with quantile paths*: one path per quantile level, the released strategy. Solid lines are geometric means over datasets, while faint points are datasets. Vertical lines separate the 1024-step rollout iterations. The first iteration is the same forward pass under the baseline, the one pass, and both rollouts.

Table 10. Mean absolute error (MW) for different TSFMs on the original skforecast Victoria electricity demand benchmark. The year 2014 is present in GIFT-Eval pretraining corpus, leading to data leakage for most TSFMs.

Model	Covariates	Point	MAE ↓
TS-ICL	✓	median	**139.93**
`t0-alpha`	✓	median	144.13
Chronos-2 (small)	✓	median	160.05
TimesFM-2.5 (200M)	✗	mean	188.28
TabPFN-TS	✓	median	190.97
Nori	✓	mean	193.33
Moirai-2.0-R (small)	✗	median	196.82
TabICL	✓	mean	210.75
Seasonal naive (24 h)	✗	—	330.59

and the `elecdemand` datasets both contain the 2014 data of the backtest [27]. To avoid this leakage, we rerun the backtest on the year 2024. Every real series of the corpus ends before 2022, so the 2024 window is free of leakage. It is at least guaranteed that `t0-alpha` has never seen the data during training.

We assemble the three variates from public sources, in the layout used by the guide. Demand is the five-minute total demand of the VIC1 region published by the Australian Energy Market Operator (AEMO) [1], averaged to hourly means. Temperature is the hourly observation of the Melbourne (Olympic Park) station, the successor of the Melbourne Regional Office site that tsibbledata used, taken from the Integrated Surface Database of the US National Centers for Environmental Information [48]. The station covers 97% of the hours. We fill the remaining gaps from the ERA5 reanalysis served by Open-Meteo [76]. The holiday indicator marks the 13 Victorian public holidays of 2024, taken from the `holidays` Python package [69]. We run the script of the guide unchanged except for the dates: 62 one-day windows from October 31 to December 31, 2024. Each model runs at two context lengths, 512 and 8192 hours, roughly three weeks and eleven months of history. All runs use an Apple M4 Pro, with accelerator hardware whenever it can be used by the model, otherwise on CPU only. Runtime is the wall-clock time of one pass over the 62 forecasts of 24 hours each, produced by the backtesting routine of skforecast. Weight loading and other overhead costs are excluded. Nori did not complete the 8192-context pass within one hour, and TabICL needed 25 minutes for it.

Table 11 reports the error at both contexts, the runtime of one pass, and the hardware. The error of every model is 2 to 2.6 times its value on the 2014 window. Leakage alone does not explain the gap. Indeed, the 2024 window is also harder, since a 24-hour seasonal naive forecast scores 614.6 MW on it against 330.6 MW on the 2014 window. Chronos-2 leads at both contexts. `t0-alpha` remains among the most accurate models while being one of the fastest. `t0-beta` places second at both contexts, and the gap to `t0-alpha` widens with the context it is given: 2.8% lower error at 512 hours and 10.1% lower at 8192, where it closes most of the distance to Chronos-2.

Table 11. Mean absolute error (MW) for different TSFMs following the skforecast Victoria electricity demand benchmark for the year 2024. The year 2024 is absent from the GIFT-Eval pretraining corpus. The best value of each error column is in bold.

		MAE ↓		Runtime (s)		
Model	**Cov.**	**512**	**8192**	**512**	**8192**	**Hardware**
Chronos-2	✓	**348.87**	**324.24**	1.1	10.5	GPU
`t0-beta`	✓	368.89	334.92	3.8	9.9	GPU
Chronos-2 (small)	✓	369.38	342.62	0.6	2.9	GPU
TS-ICL	✓	358.19	365.35	2.7	25.2	CPU
`t0-alpha`	✓	379.58	372.51	1.2	4.5	GPU
TabICL	✓	463.10	394.17	42.3	1513.3	CPU
Moirai-2.0-R (small)	✗	456.87	433.45	0.3	1.2	CPU
TimesFM-2.5 (200M)	✗	476.61	453.73	3.4	13.1	CPU
Nori	✓	388.60	—*	54.2	—*	GPU
Seasonal naive (24 h)	✗	614.57	614.57	—	—	

* Not completed within one hour on this hardware.

5.6.2 Forecasting and trading electricity in Texas

This Section 5.6.2 has been written by Macrocosm,[2] which conducted an independent evaluation of t_0 models.

[2] Macrocosm is developing simulation-driven world models, starting with the US electric grid: `mcsm.ai`.

We evaluate point and probabilistic forecasts of hourly real-time (SCED) electricity prices in the four ERCOT (the electricity operator of Texas) load zones, from January 2024 to May 2026, and assess their downstream trading value. We report MAE, CRPS and interval coverage. Each model forecasts the next operating day at 09:00 Central Time on the preceding day, using only information available at that gate, including 21 covariates covering load, renewable generation, outages and weather. As a baseline, we repeat the hourly price profile of the most recent complete operating day available at the forecast gate.

For the downstream trading evaluation, we compare five rules: three that map the predicted median spread against the day-ahead price to positions using step, linear or sigmoid functions, a Kelly rule based on the forecast distribution, and a constant-volatility rule that scales positions inversely with forecast uncertainty. For each model, we select the rule with the highest annualized daily dollar Sharpe over the evaluation window. Positions depend on the predicted spread against the day-ahead market (DAM) price and, where applicable, uncertainty. Strategy selection is retrospective. Trading uses a fixed 100 MW reference size per zone, without reinvestment or market impact. Sharpe is $\sqrt{365}$ times mean daily profit divided by its standard deviation. An always-short control sells day-ahead every hour without using forecasts.

Table 12. Forecast and trading results. MAE and CRPS are in $/MWh; coverage is for the nominal 80% interval. Trading results use each model's best rule on the same window. TimesFM-3 has one fewer forecast day.

Model	MAE	CRPS	Coverage	Profit ($M)	Sharpe
TimesFM-3	**13.78**	**10.94**	0.82	**16.42**	**1.47**
`t0-alpha`	14.55	11.50	0.74	15.00	1.34
`t0-beta`	14.46	11.37	0.75	14.94	1.29
Lagged price	23.57	—	—	14.11	1.27
Chronos-2	13.90	11.11	0.75	14.62	1.25
TiRex-2	15.22	12.08	0.76	11.21	0.97
TimesFM-2.5	16.76	13.26	0.81	11.53	0.97
Always short	—	—	—	11.82	0.96

All pretrained models have lower MAE than the lagged baseline (Table 12). TimesFM-3 has the lowest MAE and CRPS. `t0-beta` scores better than `t0-alpha` on both measures, but `t0-alpha` has a higher trading Sharpe. The two t_0 models' 80% prediction intervals contain only 74–75% of observed prices: they cover some peaks but miss extreme spikes (Figure 15). Trading profits may also reflect persistent differences between SCED and DAM prices: even the always-short rule makes money without using forecasts. Correcting forecast bias using the previous 30 days leaves the trading ranking unchanged.

6 Limitations

`t0-alpha` is the first release of the t_0 family. This section states its known limits, which serve as outlines for future directions.

Limited quantile levels The model emits five native quantile levels, from 0.1 to 0.9. A requested level that falls between two of them is interpolated, and a level outside the range returns the nearest native one, so a request for the 0.99 quantile returns the 0.9 value. The grid is nonetheless fine enough for the model to stay calibrated inside its range (Section 5.2.1), and post-hoc IQF tails recover most of what is lost outside it at no training cost

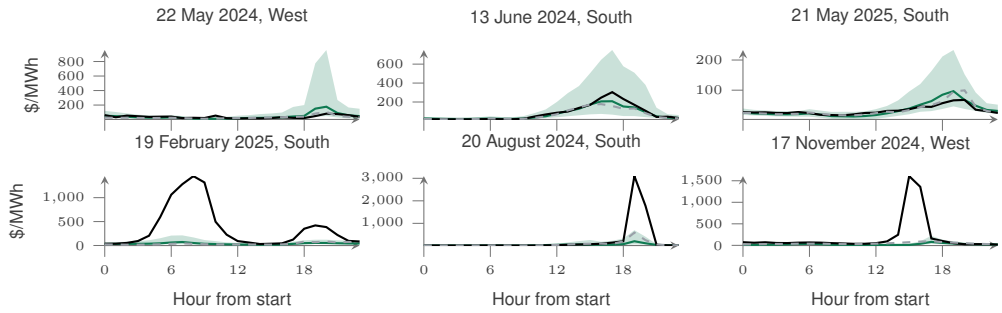

Figure 15. `t0-beta` uncertainty on six illustrative days: three peaks inside the 10–90% band (top) and three missed spikes (bottom). Green: forecast median and band; black: realized price; dashed grey: day-ahead price. Vertical scales differ; none is clipped.

(Section 5.2.2). A denser native grid, as Chronos-2 uses [6], is the remedy on the training side. `t0-beta` already emits 21 native levels, from 0.01 to 0.99.

Limited support for long lead times Recall the retail problem of Figure 1: gloves and flip-flops are manufactured and shipped months before they reach a shelf, so an apparel buyer commits the winter order in spring, six to nine months ahead. The buyer therefore needs the demand of the coming winter forecast from a context that stops at the order date, leaving a gap between the end of the context and the first forecast step. We call this gap the *lead time*. Such lead times are common in production and procurement, where decisions must often be made well before their outcomes are observed. In principle, the model naturally supports such use cases: at inference time, the gap can be represented by blanking the corresponding portion of the context, a pattern the model encounters during training through contiguous patch masking. However, the maximum lead time that can be handled this way is 1024 time steps minus one patch (32 time steps for `t0-alpha`). Beyond this limit, one must resort to autoregressive rollout, as described in Section 5.5.

Extend covariate lift studies and benchmark We measure on fev-bench the lift that covariates bring, but the benchmark does not separate covariate skill from univariate accuracy (Section 3). Two questions therefore stay open. Does the model use every covariate that carries information? Does it ignore the ones that carry none? Answering these questions calls for benchmarks built for the purpose.

Support for numerical covariates only The model conditions on past and known-future covariates as long as they are numeric series. A user who holds text, images, categorical metadata, or static attributes has to encode them as numbers first. Embedding the text description of a variate would instead let one model learn across series what a covariate means, the route taken by metadata-conditioned forecasters [20, 22] and by the multimodal models that read text alongside the series [70, 74].

This would help address cold-start predictions. A product about to launch has no past of its own. It has a photograph, a title, a category, and a description, and the products already on the shelf carry those attributes together with their sales. A model that read all of it could place the new item among the old ones and forecast from that neighborhood. No purely numeric method can do so, and `t0-alpha` is one.

No joint distribution The model emits marginal quantiles, so its output ties neither the steps of a horizon nor the variates of a sample together (Section 2).

No controlled scaling study This report evaluates the 102M-parameter `t0-alpha` model in depth and reports public-benchmark results for its 256M-parameter successor, `t0-beta`. It does not study the scaling of the t_0

family. It does not isolate the effect of parameter count from other differences between the models and their evaluation protocols. Across model families, parameter count alone does not yet determine accuracy. TimesFM-3.0 achieves a GIFT-Eval score of 0.4557 CRPS with 331M parameters, compared with 0.4759 for the 2.5B-parameter Toto-2.0 model (Figure 2). Within the Toto-2.0 family, increasing model size improves accuracy, but the gain from 313M to 2.5B parameters is comparatively small, from 0.4814 to 0.4759 CRPS. Another model, Timer-S1, with 8.3B parameters, 0.75B of them activated per token, scores 0.4853 [41]. Xihe-max with 1.5B parameters scores 0.4905 [67]. Both models sit behind TimesFM-3.0 on GIFT-Eval, a model less than a quarter of their size. Establishing how t_0 benefits from additional model capacity, training data, and compute requires controlled experiments.

7 Conclusion

We presented `t0-alpha`, a 102M-parameter open-weights forecasting model. Factorized time/variate attention gives target series, past covariates, and known-future covariates one representation and one forward pass. Pre-training combines curated public corpora with synthetic families whose covariate-to-target structure is known by construction. We report accuracy on GIFT-Eval and fev-bench together with the covariate lift, calibration and tail behavior, rank stability across eleven metrics, robustness to missing history, and context efficiency (Section 5). Passing covariates to the same checkpoint raises fev-bench skill by 6.3 points on the tasks with known-future covariates and 2.7 points on those with past covariates only (Section 5.1.2). The weights (Apache-2.0), the `tfc-t0` package, and a notebook reproducing the GIFT-Eval results are public.

`t0-beta`, the successor of `t0-alpha`, already ranks among the strongest zero-shot models we compare. On the same three public benchmarks it reaches 0.4738 CRPS on GIFT-Eval, third among those models and 1.4% short of second place, 46.4 skill on fev-bench, third again and 46.1 on its covariate-informed half, and 0.5551 CRPS on TIME (Section 5.1).

`t0-alpha` is the first release of the t_0 family. The open cases of Section 6 set the agenda for the next iterations.

References

[1] AEMO. Aggregated price and demand data, region VIC1, 2023–2024. Australian Energy Market Operator, monthly CSV files `PRICE_AND_DEMAND_<YYYYMM>_VIC1.csv` at https://aemo.com.au/aemo/data/nem/priceanddemand/, 2024. https://aemo.com.au/energy-systems/electricity/national-electricity-market-nem/data-nem/aggregated-data. Five-minute `TOTALDEMAND` in MW on National Electricity Market time. AEMO permits use of its material for any purpose with attribution (https://aemo.com.au/privacy-and-legal-notices/copyright-permissions); accessed September 8, 2026.

[2] 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. arxiv/2410.10393. Also presented at the NeurIPS 2024 Workshop on Time Series in the Age of Large Models.

[3] Joaquin Amat Rodrigo and Javier Escobar Ortiz. skforecast. Version 0.24.0, Zenodo, 2026. doi:10.5281/zenodo.8382788. Python library for time series forecasting with a scikit-learn interface, BSD-3-Clause, https://skforecast.org/.

[4] Joaquin Amat Rodrigo and Javier Escobar Ortiz. Forecasting with foundation models. skforecast user guide, 2026. https://skforecast.org/latest/user_guides/

foundation-forecasting-models. Also published at https://cienciadedatos.net/documentos/py79-forecasting-with-foundation-models.html; accessed September 7, 2026.

[5] 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. *Transactions on Machine Learning Research*, 2024. arxiv/2403.07815.

[6] 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. *arXiv preprint arXiv:2510.15821*, 2025. arxiv/2510.15821.

[7] Andreas Auer, Raghul Parthipan, Pedro Mercado, Abdul Fatir Ansari, Lorenzo Stella, Bernie Wang, Michael Bohlke-Schneider, and Syama Sundar Rangapuram. Zero-shot time series forecasting with covariates via in-context learning. *arXiv preprint arXiv:2506.03128*, 2025. arxiv/2506.03128.

[8] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arxiv/2505.23719.

[9] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arxiv/2405.04517.

[10] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, 2009. doi:10.1145/1553374.1553380.

[11] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day, San Francisco, 1970.

[12] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. Wiley, Hoboken, NJ, 5th edition, 2015. Seasonal (SARIMA) models.

[13] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 785–794, 2016. doi:10.1145/2939672.2939785.

[14] Victor Chernozhukov, Iván Fernández-Val, and Alfred Galichon. Quantile and probability curves without crossing. *Econometrica*, 78(3):1093–1125, 2010. doi:10.3982/ECTA7880.

[15] Ben Cohen, Emaad Khwaja, Kan Wang, Charles Masson, Elise Ramé, Youssef Doubli, and Othmane Abou-Amal. Toto: Time series optimized transformer for observability. *arXiv preprint arXiv:2407.07874*, 2024. arxiv/2407.07874.

[16] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning (ICML)*, 2024. arxiv/2310.10688.

[17] Abhimanyu Das, Matthew Faw, Rajat Sen, and Yichen Zhou. In-context fine-tuning for time-series foundation models. In *International Conference on Machine Learning (ICML)*, 2025. arxiv/2410.24087.

[18] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

arxiv/2302.05442.

[19] Hantian Ding, Zijian Wang, Giovanni Paolini, Varun Kumar, Anoop Deoras, Dan Roth, and Stefano Soatto. Fewer truncations improve language modeling. In *International Conference on Machine Learning (ICML)*, 2024. arxiv/2404.10830.

[20] Jiaxiang Dong, Haixu Wu, Yuxuan Wang, Li Zhang, Jianmin Wang, and Mingsheng Long. Metadata matters for time series: Informative forecasting with transformers. *arXiv preprint arXiv:2410.03806*, 2024. arxiv/2410.03806. Encodes dataset- and variate-level descriptions as LLM metadata tokens alongside series tokens.

[21] Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha V. Naidu, and Colin White. Forecastpfn: Synthetically-trained zero-shot forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arxiv/2311.01933.

[22] Utsav Dutta, Sina Khoshfetrat Pakazad, and Henrik Ohlsson. Time to embed: Unlocking foundation models for time series with channel descriptions. *arXiv preprint arXiv:2505.14543*, 2025. arxiv/2505.14543. CHARM; channel-level textual descriptions, invariant to channel order.

[23] Carson Eisenach, Yagna Patel, and Dhruv Madeka. Mqtransformer: Multi-horizon forecasts with context dependent and feedback-aware attention. *arXiv preprint arXiv:2009.14799*, 2020. arxiv/2009.14799.

[24] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. doi:10.1214/aos/1013203451.

[25] Tilmann Gneiting and Roopesh Ranjan. Comparing density forecasts using threshold- and quantile-weighted scoring rules. *Journal of Business & Economic Statistics*, 29(3):411–422, 2011. doi:10.1198/jbes.2010.08110.

[26] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007. doi:10.1111/j.1467-9868.2007.00587.x.

[27] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey Webb, Rob Hyndman, and Pablo Montero-Manso. Australian electricity demand dataset. Monash Time Series Forecasting Archive, Zenodo, 2021. doi:10.5281/zenodo.4659727. Half-hourly demand of five Australian states, extracted from the tsibbledata R package.

[28] Google Research. Timesfm 3.0. Hugging Face, 2026. https://huggingface.co/google/timesfm-3.0-pytorch. Model release; no standalone paper. The release directs citations to arXiv:2310.10688, which describes the original TimesFM architecture and not this one.

[29] Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4246–4253, 2020. doi:10.18653/v1/2020.findings-emnlp.379. arxiv/2010.04245.

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

[31] Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. From tables to time: Extending TabPFN-v2 to time series forecasting. *Transactions on Machine Learning Research*, 2026. arxiv/2501.02945. Circulated in 2025 as "The Tabular Foundation Model TabPFN Outperforms Specialized Time Series Forecasting Models Based on Simple Features"; the TMLR version is substantially rewritten.

[32] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Melbourne, Australia, 3rd edition, 2021. https://otexts.com/fpp3/.

[33] Rob J. Hyndman, Anne B. Koehler, J. Keith Ord, and Ralph D. Snyder. *Forecasting with Exponential Smoothing: The State Space Approach*. Springer Series in Statistics. Springer, Berlin, 2008. doi:10.1007/978-3-540-71918-2.

[34] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3146–3154, 2017. https://papers.nips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

[35] 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. *arXiv preprint arXiv:2605.20119*, 2026. arxiv/2605.20119.

[36] Roger Koenker and Gilbert Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978. doi:10.2307/1913643.

[37] Mario Michael Krell, Matej Kosec, Sergio P. Perez, and Andrew Fitzgibbon. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *arXiv preprint arXiv:2107.02027*, 2021. arxiv/2107.02027.

[38] Francesco Laio and Stefania Tamea. Verification tools for probabilistic forecasts of continuous hydrological variables. *Hydrology and Earth System Sciences*, 11(4):1267–1277, 2007. doi:10.5194/hess-11-1267-2007.

[39] Etienne Le Naour, Tahar Nabil, and Adrien Petralia. TS-ICL: A flexible time-indexed foundation model for time series via in-context learning. *arXiv preprint arXiv:2606.05878*, 2026. arxiv/2606.05878.

[40] Yiding Liu, Yifan Hu, Hongjie Xia, Peiyuan Liu, Hongzhou Chen, Xilin Dai, Zewei Dong, and Jiang-Ming Yang. Falcon-X: A time series foundation model for heterogeneous multivariate modeling. *arXiv preprint arXiv:2605.27286*, 2026. arxiv/2605.27286.

[41] 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. *arXiv preprint arXiv:2603.04791*, 2026. arxiv/2603.04791. Tsinghua and ByteDance; 8.3B total parameters, 0.75B activated per token.

[42] Andrea Lodi, Silvano Martello, and Michele Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, 2002. doi:10.1016/S0377-2217(02)00123-6.

[43] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. arxiv/1711.05101.

[44] Helmut Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer, Berlin, 2005. doi:10.1007/978-3-540-27752-1.

[45] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022. doi:10.1016/j.ijforecast.2021.11.013.

[46] Vladyslav Moroshan, Julien Siems, Arber Zela, Timur Carstensen, and Frank Hutter. Tempopfn: Synthetic pre-training of linear rnns for zero-shot time series forecasting. *arXiv preprint arXiv:2510.25502*, 2025. arxiv/2510.25502.

[47] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations (ICLR)*, 2023. arxiv/2211.14730.

[48] NOAA NCEI. Global surface hourly [Integrated Surface Database, station 959360-99999, Melbourne (Olympic Park), 2023–2024]. NOAA National Centers for Environmental Information, 2001. https://www.ncei.noaa.gov/data/global-hourly/. Non-US stations fall under WMO Resolution 40: free for research, education and other non-commercial use; the data and derived products may not be redistributed. Accessed September 8, 2026.

[49] Mitchell O'Hara-Wild, Rob Hyndman, Earo Wang, and Rakshitha Godahewa. tsibbledata: Diverse datasets for 'tsibble'. R package, 2022. https://tsibbledata.tidyverts.org/. Dataset vic_elec: half-hourly electricity demand for Victoria, Australia.

[50] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2020. arxiv/1905.10437.

[51] Alan Pankratz. *Forecasting with Dynamic Regression Models*. Wiley Series in Probability and Statistics. Wiley, New York, 1991. doi:10.1002/9781118150528.

[52] Youngsuk Park, Danielle Maddix, François-Xavier Aubet, Kelvin Kan, Jan Gasthaus, and Yuyang Wang. Learning quantile functions without quantile crossing for distribution-free time series forecasting. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 151 of *Proceedings of Machine Learning Research*, pages 8127–8150, 2022. arxiv/2111.06581.

[53] Sebastian Pineda Arango, Pedro Mercado, Shubham Kapoor, Abdul Fatir Ansari, Lorenzo Stella, Huibin Shen, Hugo Senetaire, Caner Turkmen, Oleksandr Shchur, Danielle C. Maddix, Michael Bohlke-Schneider, Yuyang Wang, and Syama Sundar Rangapuram. Chronosx: Adapting pretrained time series models with exogenous variables. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2025. arxiv/2503.12107.

[54] Patrick Podest, Marco Pichler, Elias Bürger, Levente Zó lyomi, Bernhard Voggenberger, Wilhelm Berghammer, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Tirex-2: Generalizing tirex to multivariate data and streaming. *arXiv preprint arXiv:2607.01204*, 2026. arxiv/2607.01204.

[55] Andres Potapczynski, Ravi Kiran Selvam, Tatiana Konstantinova, Shankar Ramasubramanian, Malcolm Wolff, Kin G Olivares, Ruijun Ma, Mengfei Cao, Michael W Mahoney, Andrew Gordon Wilson, et al. Time-aware prior fitted networks for zero-shot forecasting with exogenous variables. *arXiv preprint arXiv:2603.15802*, 2026. https://arxiv.org/pdf/2603.15802.

[56] Willa Potosnak, Malcolm Wolff, Mengfei Cao, Ruijun Ma, Tatiana Konstantinova, Dmitry Efimov, Michael W. Mahoney, Boris Oreshkin, and Kin G. Olivares. Forking-sequences: Statistically and computationally efficient multi-horizon forecasting with reduced volatility. *Transactions on Machine Learning Research*, 2026. arxiv/2510.04487.

[57] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. arxiv/1706.09516.

[58] 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. In *International Conference on Machine Learning (ICML)*, 2026. arxiv/2602.12147.

[59] Guo Qin, Zhi Chen, Yong Liu, Zhiyuan Shi, Haixuan Liu, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Cora: Covariate-aware adaptation of time series foundation models. *arXiv preprint arXiv:2510.12681*, 2025. arxiv/2510.12681.

[60] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. arxiv/1910.10683.

[61] Francesca Rochberg. *The Heavenly Writing: Divination, Horoscopy, and Astronomy in Mesopotamian Culture*. Cambridge University Press, Cambridge, 2004. doi:10.1017/CBO9780511617409.

[62] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020. doi:10.1016/j.ijforecast.2019.07.001.

[63] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. arxiv/2002.05202.

[64] Oleksandr Shchur, Abdul Fatir Ansari, Caner Turkmen, Lorenzo Stella, Nick Erickson, Pablo Guerron, Michael Bohlke-Schneider, and Yuyang Wang. fev-bench: A realistic benchmark for time series forecasting. *arXiv preprint arXiv:2509.26468*, 2025. arxiv/2509.26468.

[65] John M. Steele. Eclipse prediction in mesopotamia. *Archive for History of Exact Sciences*, 54(5):421–454, 2000. doi:10.1007/PL00021245.

[66] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi:10.1016/j.neucom.2023.127063. arxiv/2104.09864.

[67] Yinbo Sun, Yuchen Fang, Zhibo Zhu, Jia Li, Yu Liu, Qiwen Deng, Jun Zhou, Hang Yu, Xingyu Lu, and Lintao Ma. Xihe: Scalable zero-shot time series learner via hierarchical interleaved block attention. *arXiv preprint arXiv:2510.21795*, 2025. arxiv/2510.21795. Ant Group; sizes from 9.5M (tiny) to 1.5B (max).

[68] Yutao Sun, Li Dong, Barun Patra, Shuming Ma, Shaohan Huang, Alon Benhaim, Vishrav Chaudhary, Xia Song, and Furu Wei. A length-extrapolatable transformer. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 14590–14604, 2023. doi:10.18653/v1/2023.acl-long.816. arxiv/2212.10554.

[69] Vacanza Team. holidays: Open World Holidays Framework. Python package, 2026. https://github.com/vacanza/holidays. MIT license.

[70] Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, and Jianxin Liao. Chattime: A unified multimodal time series foundation model bridging numerical and textual data. In *AAAI Conference on Artificial Intelligence*, 2025. arxiv/2412.11376.

[71] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017. arxiv/1711.11053. NIPS 2017 Time Series Workshop.

[72] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *International Conference on Machine Learning (ICML)*, 2024. arxiv/2402.02592. Introduces the LOTSA corpus.

[73] 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. In *International Conference on Learning Representations (ICLR)*, 2026. arxiv/2508.02879. Oral presentation.

[74] Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Dan Pei, et al. Chatts: Aligning time series with LLMs via synthetic data for enhanced understanding and reasoning. *Proceedings of the VLDB Endowment*, 18(8):2385–2398, 2025. doi:10.14778/3742728.3742735. arxiv/2412.03104.

[75] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. arxiv/1910.07467.

[76] Patrick Zippenfenig. Open-Meteo.com Weather API. Zenodo, 2023. doi:10.5281/zenodo.7970649. Historical weather API serving the ERA5 reanalysis, https://open-meteo.com/en/docs/

historical-weather-api. Weather data by Open-Meteo.com, CC BY 4.0; accessed September 8, 2026.

Acknowledgments

The Forecasting Company and the authors thank Macrocosm for agreeing to evaluate the t_0 models independently, and for the time and effort that took. Macrocosm designed the ERCOT forecasting and trading study of Section 5.6.2, ran it on their own data and infrastructure, and wrote the section.

A GIFT-Eval cross-domain subset

Table 13 lists the twelve dataset and term pairs that the studies of Sections 5.2.2, 5.3 and 5.4 run on. The twelve pairs come from twelve distinct datasets, one pair each. They cover all seven GIFT-Eval domains, four frequencies from 15-minute to monthly, and the three horizon terms, at a fifth of the inference cost per configuration of the full 97-pair benchmark. Two of the twelve carry irregularity by design: kdd_cup_2018 has native gaps and car_parts is intermittent, so the missing-data study of Section 5.3 measures injected masks on top of missingness a user would also meet.

Table 13. The twelve GIFT-Eval dataset and term pairs used by the studies of Sections 5.2.2, 5.3 and 5.4. Dataset names, frequency codes and terms are those GIFT-Eval declares; domains are its seven leaderboard classes.

Dataset	Frequency	Term	Domain
electricity	D	short	Energy
solar	H	long	Energy
bizitobs_l2c	H	short	Web/CloudOps
bitbrains_rnd	H	short	Web/CloudOps
loop_seattle	D	short	Transport
sz_taxi	15T	long	Transport
jena_weather	H	short	Nature
kdd_cup_2018	H	medium	Nature
restaurant	D	short	Sales
car_parts	M	short	Sales
m4_hourly	H	short	Econ/Fin
covid_deaths	D	short	Healthcare