SPECTOR: SPECTRAL TREE OPERATOR REALIZATION FOR TIME-SERIES PRIORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series foundation models increasingly rely on synthetic data to supplement limited real-world corpora and to improve the scale and diversity of pretraining. Yet existing generators face a trade-off: compositional kernels offer rich diversity but require cubic-time covariance factorization, while fast parametric simulators scale linearly at the cost of narrower structure and weaker transfer. We introduce SPECTOR (**SPEC**tral **T**ree **O**perator **R**ealization), a realization compiler for compositional time-series priors that resolves this trade-off. SPECTOR treats a sum-product kernel expression as a stochastic-process program and recursively compiles its subtrees into efficient realization spaces, without materializing covariance matrices while enabling structural diversity and linear-time sampling. We formalize the semantics of this compiler through a covariance-preserving result for general realization trees, with spectral and feature-space semiring constructions providing stronger law-preserving guarantees for important kernel classes. With 18 kernel primitives, SPECTOR spans stationary, periodic, trending, transient, and stochastic-drift behaviors. Empirically, SPECTOR scales to long contexts and is hundreds of times faster than covariance-based kernel generators, enabling on-the-fly pretraining. Across six architectures and three benchmarks, SPECTOR achieves 74–91% pairwise win rates against competing synthetic priors and narrows the gap to published models pretrained on hybrid real-synthetic corpora.

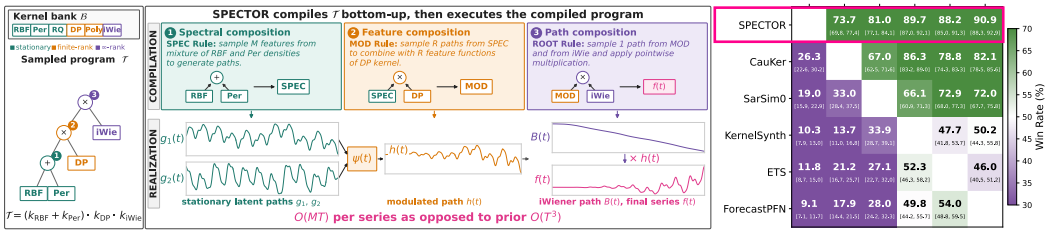
1 INTRODUCTION

Zero-shot time-series foundation models (TSFMs), used with frozen parameters and no target-data adaptation, rely on broad pretraining distributions to transfer across domains, resolutions and seasonalities. Large collections of real-world time series provide one route to constructing broad pretraining distributions. However, such collections remain small relative to public text corpora and are difficult to scale due to licensing, privacy, temporal leakage, and train/test contamination. Recent work has therefore increasingly incorporated synthetic series into TSFM pretraining (Ansari et al., 2025; Khwaja et al., 2026; Wen et al., 2026), in several cases training models entirely from simulated data (Dooley et al., 2023; Xie et al., 2026; Moroshan et al., 2025; Oreshkin et al., 2026).

As synthetic data takes on a larger share of the pretraining distribution, its role extends beyond that of augmenting real-world data. When examples are generated on the fly, the training distribution can be specified programmatically and instantiated only as needed during optimization, without first constructing or storing a fixed corpus. We call this approach *in-situ pretraining*, borrowing from high performance computing (HPC), where *in-situ processing* operates on simulation data as they are generated (Bauer et al., 2016). Adopting in-situ pretraining can simplify model provenance and deployment review, particularly under synthetic-only pretraining, where data lineage need not be traced across large collections of proprietary or restricted datasets. The (implicitly-defined) pretraining distribution also becomes portable and versionable as code, while retaining a clear record of how training examples are generated. Model improvement can then proceed without necessarily collecting additional real-world data: observed failure modes can be addressed by modifying the (implicitly-defined) prior, generating targeted examples, and iterating programmatically, including through agentic development loops.

For *in-situ pretraining* to be practical, the generative prior must satisfy two requirements simultaneously. First, it must be sufficiently expressive to cover a broad range of temporal structure. Second, the prior must generate examples as fast as pretraining consumes them. Compositional ker-

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



(a) SPECTOR Overview **(b)** Pairwise Forecasting Win Rate

Figure 1: **(a)** SPECTOR compiles compositional generator programs into efficient linear-in-length time-series samplers using *spectral composition*, (Thm. 2.3), *feature composition* (Prop. 2.5), and *path composition* (Prop. 2.1). **(b)** Pretraining on SPECTOR yields 74–91% pairwise win rates against other generators across 6 models and 210 zero-shot forecasting tasks.

nel priors provide a particularly attractive route to achieving the first requirement (Duvenaud et al., 2013) since a compact primitive vocabulary can generate a combinatorially large family of processes through combination. KernelSynth (Ansari et al., 2024), a commonly used synthetic generator for various TSFMs like Chronos and Tirex Auer et al. (2026), reverses the Automatic Statistician (Lloyd et al., 2014): instead of searching over kernel compositions to explain data, it samples them to generate diverse stochastic processes. However, its exact Gaussian process (GP) sampling incurs cubic time and quadratic memory through dense covariance factorization, making naïve compositional prior realization impractical for in-situ pretraining. In contrast, parametric synthetic generators like ETS (Hyndman et al., 2008) and SarSim0 (Oreshkin et al., 2026) are efficient, satisfying the second requirement, but represent a more restricted family of temporal structures. We refer readers to Appendix B for further discussion of existing synthetic data generators for TSFMs and their challenges.

In this paper, we address this expressivity-efficiency tradeoff by moving from covariance-based sampling to direct realization-space sampling of compositional stochastic-process priors. We make three main contributions. (1) **A realization compiler for compositional stochastic-process programs.** We treat a sum-product kernel expression as an executable prior program and recursively compile its subtrees into efficient realization spaces: spectral for stationary structure and feature-space for finite-rank structure; direct recursive or modulation-based realizations otherwise. We show that this execution preserves the kernel semantics of the original program, with stronger law-preserving guarantees for spectral and feature-space subclasses. The approach is extensible: new primitives require only a realization and compatible composition rules, not new covariance machinery. (2) **SPECTOR, a scalable instantiation of this compiler.** SPECTOR (Figure 1a) combines 18 stochastic primitives spanning stationary, periodic, trending, transient, and stochastic-drift behaviors with Fourier-feature, feature-space, and direct time-domain sampling and augmentations. This instantiation retains broad compositional structure while scaling linearly with sequence length, making long-context and in-situ pretraining practical. (3) **Strong synthetic-only generalization across architectures and benchmarks.** Across six forecasting architectures and 210 zero-shot tasks, SPECTOR consistently outperforms competing synthetic priors, matches or exceeds mixtures of multiple generators, and substantially narrows the gap to models pretrained on hybrid real-synthetic corpora.

2 SPECTOR

In this paper, we focus on univariate forecasting. Given an observed context $\mathbf{x}^{(i)}$, a forecasting model f_θ predicts quantiles $\hat{Q}^{(i)}_{t,q}$ for supervised targets $y^{(i)}_t$, where $q \in \mathcal{Q}$ indexes fixed quantile levels and $(i,t) \in \mathcal{I}$ indexes supervised prediction positions. This notation covers both standard horizon forecasting and masked-position objectives. Following the zero-shot protocol, θ is frozen at inference, and f_θ is applied independently to *one series at a time*, without target-data adaptation (Olivares et al., 2025).

We now formalize SPECTOR as a compiler for compositional stochastic-process programs, showing how efficient kernel realizations can be composed recursively while preserving the semantics of an arbitrary sum-product kernel expression. We view a kernel expression as an *algebraic program*: primitive kernels form the leaves, while sums and products define its operators. We first establish a general covariance-consistency result independent of any particular representation, then derive spectral and feature-space semiring structures that enable stronger law-preserving compilation. These

results underlie SPECTOR, which recursively dispatches each subtree to its most efficient available realization: spectral, feature-space, or direct time-domain, without dense covariance construction or factorization. We then describe the full SPECTOR generation pipeline, including the expanded kernel bank, sampling procedure, and augmentations. To streamline the presentation, we compactly present theoretical results and leave references and proofs to Appendix B and D.

Realization Theory for Compositional Kernel Programs. Let $\mathcal{B}$ be a bank of primitive positive-semidefinite kernels, and let $\mathcal{T}$ denote any finite expression formed from elements of $\mathcal{B}$ using kernel addition and pointwise multiplication, as illustrated in Figure 1. Since positive-semidefinite kernels are closed under these operations, their evaluations lie in a commutative semiring (Berlinet & Thomas-Agnan, 2012). We write $k_{\mathcal{T}} = \mathrm{Eval}_K(\mathcal{T})$ for the resulting kernel. The syntax $\mathcal{T}$ specifies how primitives are composed; its kernel semantics is $\mathrm{Eval}_K(\mathcal{T})$. On a finite grid, pointwise kernel multiplication corresponds to the Hadamard product of the associated Gram matrices (Grenander et al., 2013). Examples and remarks on singular and degenerate kernels are provided in Appendix C.

We now distinguish this kernel semantics from its stochastic realization: $\mathcal{B}$ contains covariance kernels, whereas $\mathcal{R}$ contains random processes whose covariances realize those kernels. Let $\mathcal{R}$ denote the set of zero-mean second-order stochastic processes on the index set: random functions f with $\mathbb{E}[f(t)] = 0$ and $\mathbb{E}[f(t)^2] < \infty$ for every t. Each primitive-kernel occurrence at a leaf of the expression tree, $k_\ell \in \mathcal{B}$, is associated with an independent $f_\ell \in \mathcal{R}$ satisfying $\mathrm{Cov}\big(f_\ell(t), f_\ell(t')\big) = k_\ell(t, t')$. We then define realization composition by pointwise path addition and multiplication:

$$(f_a \oplus f_b)(t) = f_a(t) + f_b(t), \qquad (f_a \odot f_b)(t) = f_a(t) f_b(t).$$

For *independent* $f_a, f_b \in \mathcal{R}$ with covariance kernels k_a and k_b, these operations reproduce kernel addition and multiplication at the covariance level: $\mathrm{Cov}[f_a \oplus f_b] = k_a + k_b$, $\mathrm{Cov}[f_a \odot f_b] = k_a k_b$. Based on this, we have the following result, which is proven in Appendix D.1.

Proposition 2.1 (Covariance consistency of realization trees)**.** *Let $\mathcal{T}$ be any finite sum-product expression whose leaf occurrences are assigned independent zero-mean realizations with the corresponding covariance kernels. If $\mathrm{Eval}_R(\mathcal{T})$ evaluates the same expression using $\oplus$ and $\odot$ in realization space, then $\mathrm{Eval}_R(\mathcal{T}) \in \mathcal{R}$ and $\mathrm{Cov}[\mathrm{Eval}_R(\mathcal{T})] = \mathrm{Eval}_K(\mathcal{T})$.*

The significance of Proposition 2.1 is algorithmic: it lifts local realizations of primitive kernels to arbitrary finite sum-product expression trees. Appendix C.3 formalizes why this realization semantics is distinct from a semiring homomorphism. Consequently, a composite prior can be executed recursively in realization space without first materializing the composite kernel or its covariance matrix. Appendix C.1 gives a concrete example. This construction preserves covariance but need not preserve the full stochastic-process law: products of Gaussian processes are generally non-Gaussian.

Law-preserving compilation. In the following result, which is proven in Appendix D.2, we show that stronger law-preserving realizations are possible for stationary kernels.

Proposition 2.2 (Spectral isomorphism for compositional kernel trees)**.** *Let $\mathcal{K}_{\mathrm{stat}}$ be the semiring of continuous stationary positive-semidefinite kernels on $\mathbb{R}$ under addition and multiplication, and let $\mathcal{M}_+$ be the semiring of finite nonnegative symmetric measures under addition and convolution. By Bochner's theorem, the spectral map $\mathcal{S} : (\mathcal{K}_{\mathrm{stat}}, +, \cdot) \to (\mathcal{M}_+, +, *)$ is a semiring isomorphism. Hence, for any finite sum-product tree $\mathcal{T}$ with stationary leaves, $\mathcal{S}(\mathrm{Eval}_K(\mathcal{T})) = \mathrm{Eval}_{\mathcal{S}}(\mathcal{T})$.*

For SPECTOR, the key consequence is that the Fourier-domain identities implied by Bochner's theorem (Bochner, 1933; Rudin, 2017) induce compositional compiler rules over arbitrary expression trees: kernel sums remain sums, while products become spectral convolutions. For zero-mean GPs, this equivalence also preserves the process law. Appendix C.2 displays a commuting diagram.

This representation becomes computationally useful only if the composite spectral measure can be sampled without constructing it. Theorem 2.3, proven in Appendix D.3, provides exactly this step. SPECTOR turns the syntax of a stationary sum-product subtree into a recursive frequency-sampling program: sum nodes route draws to one child according to spectral mass, while product nodes add independent child frequency draws. The composite spectral density is never formed explicitly.

Theorem 2.3 (Spectral tree compilation)**.** *Let $\mathcal{T}$ be a finite sum-product tree of stationary kernels with spectral measure $\mu_{\mathcal{T}} = \mathrm{Eval}_{\mathcal{S}}(\mathcal{T})$ and $k_{\mathcal{T}}(0) > 0$. Its normalized spectral law $p_{\mathcal{T}} = d\mu_{\mathcal{T}}/k_{\mathcal{T}}(0)$ can be sampled exactly by recursively drawing a frequency $\omega_\ell \sim d\mu_\ell/k_\ell(0)$ at each leaf ℓ, selecting a child of a sum node (a, b) with probability proportional to $k_a(0)$ and $k_b(0)$,*

and returning $\omega_a + \omega_b$ from independent child draws at a product node (a, b). Thus, the resulting program $\mathcal{P}_\mathcal{T}$ samples exactly from the normalized spectral measure represented by $\mathrm{Eval}_\mathcal{S}(\mathcal{T})$.

Theorem 2.3 samples exactly from the composite spectral law. Proposition D.2 stated in Appendix D.4 turns this into a covariance-free time-domain realization via M frequency draws:

$$f_M(t) = \sqrt{\frac{2\,k(0)}{M}} \sum_{m=1}^{M} \cos(\omega_m t + b_m), \tag{1}$$

and establishes that f_M has unbiased target covariance, $\mathbb{E}[f_M(t) f_M(t')] = k(t - t')$ for all t, t'; and its finite-dimensional distributions converge to the target GP law at the rate $O(M^{-1/2})$. Algorithm 4 in Appendix A.1 gives the corresponding sampling procedure. The practical advantage is that many primitive spectral laws are simple. For example, the squared-exponential kernel $k_{\mathrm{SE}}(\tau) = \sigma^2 \exp(-\tau^2/2\ell^2)$ has spectral law $\omega \sim \mathcal{N}(0, \ell^{-2})$, so each exact leaf draw is a single Gaussian sample, independent of T. Theorem 2.3 extends this simplicity compositionally, compiling sum and product structure into a recursive program of child selection and frequency addition. Unlike random Fourier features, which assume access to a fixed spectral law, SPECTOR recursively samples the spectral law of arbitrary symbolic compositions without constructing the composite spectrum or $T \times T$ covariance matrix, enabling $O(MT)$ Fourier synthesis with M frequencies.

We next turn to nonstationary kernels, for which Bochner-based spectral compilation is generally unavailable. For the important class of *finite-rank kernels*, $k_{\mathrm{FR}}(t, t') = \psi(t)^\top \psi(t')$ with $\psi(t) \in \mathbb{R}^r$, drawing $z \sim \mathcal{N}(0, I_r)$ and setting $f(t) = \psi(t)^\top z$ gives an exact GP realization in $O(rT)$ time. This class includes several useful nonstationary trend and growth kernels; representative examples and their feature maps are given in Appendix E.1.

Feature-map composition for finite-rank kernels provides a second representation algebra: sums map to concatenation and products to tensor products, allowing the compiler abstraction to operate on nonstationary finite-rank subtrees. Proposition 2.4, proven in Appendix D.5, formalizes this.

Proposition 2.4 (Finite-rank feature homomorphism)**.** *Let $\mathcal{K}_{\mathrm{FR}}$ be the semiring of finite-rank positive-semidefinite kernels under addition and multiplication. Let $\mathcal{F}_{\mathrm{FR}}$ be the semiring of finite-dimensional feature maps under concatenation and tensor product. The map $\kappa : (\mathcal{F}_{\mathrm{FR}}, \oplus, \otimes) \to (\mathcal{K}_{\mathrm{FR}}, +, \cdot)$, defined by $\kappa([\psi])(t, t') = \psi(t)^\top \psi(t')$, is a surjective semiring homomorphism. Hence, for any finite sum-product tree $\mathcal{T}$ with finite-rank leaves, $\kappa(\mathrm{Eval}_\mathcal{F}(\mathcal{T})) = \mathrm{Eval}_K(\mathcal{T})$.*

Hence, the $O(rT)$ realization extends from individual kernels to entire finite-rank subtrees, with sums compiled by feature concatenation and products by tensor products. Thus, finite-rank subtrees inherit a representation-space algebra analogous to the spectral algebra above, allowing entire subtrees to be compiled in $\mathcal{F}_{\mathrm{FR}}$ without materializing its kernel. Under this view, spectral and finite-rank feature compilation are two instances of the same pattern. The compositional program is evaluated in a representation semiring, while semiring homomorphism: $\mathcal{S}$ from kernels to spectral measures, κ from feature representations to kernels, ties that evaluation to the kernel semantics. This perspective makes the framework extensible: incorporating a new primitive requires specifying its realization in an appropriate target space, after which the corresponding semiring operations determine its compositions. A formal free-semiring interpretation of this insight is given in Appendix C.3.

Feature-space compilation is exact, but products multiply feature dimension: if k_1 and k_2 have ranks r_1 and r_2, then $k_1 k_2$ has rank at most $r_1 r_2$ (see Corollary D.3 in Appendix D.5). Naive feature-product compilation can cause multiplicative rank growth. SPECTOR therefore uses *Gaussian modulation* for mixed finite-rank $\times$ Gaussian subtrees, replacing explicit tensor products with independent GP realizations. The following proposition, proven in Appendix D.6, formalizes this.

Proposition 2.5 (Gaussian modulation)**.** *Let $k_{\mathrm{FR}}(t, t') = \sum_{j=1}^{r} \psi_j(t)\psi_j(t')$ and let $g_j \overset{\mathrm{ind}}{\sim} \mathcal{GP}(0, k_g)$. Then $f(t) = \sum_{j=1}^{r} \psi_j(t) g_j(t)$ satisfies $f \sim \mathcal{GP}(0, k_{\mathrm{FR}} k_g)$.*

Gaussian modulation avoids tensor-product expansion by replacing the scalar Gaussian coefficients of a finite-rank representation with independent realizations of the second process, preserving both product covariance and the full GP law. If each g_j is generated using the spectral sampler with M frequencies the cost is $O(rMT)$. When both factors are finite rank, the construction reduces to the tensor-product feature representation of Proposition 2.4.

Direct time-domain realization. An important class of *recursive kernels*, useful for modeling stochastic trends and drift, lacks finite-rank representations but admits finite-dimensional linear Gaussian state-space forms and efficient recursive sampling. The Wiener kernel $k_{\mathrm{W}}(t, t') = \sigma^2 \min(t, t')$ is sampled by Gaussian increments $f(t_i) = f(t_{i-1}) + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma^2(t_i - t_{i-1}))$. An order-$n$ integrated Wiener kernel follows from n successive cumulative sums, each costing $O(T)$. More generally, finite-dimensional linear Gaussian state-space kernels admit recursive covariance-free sampling. Appendix E.2 provides further examples.

Composition in this space follows Proposition 2.1: sums are law-preserving, while products preserve only covariance. For example, if B is Wiener and g an independent stationary realization with M spectral features, then $f(t) = B(t)g(t)$ has covariance $\sigma^2 \min(t, t')\, k_{\mathrm{stat}}(t - t')$ at cost $O(MT)$. Direct time-domain realization also supports non-Gaussian leaves; for example, Student-t noise shares white noise's covariance but has heavy-tailed marginals. Since Proposition 2.1 requires only zero-mean second-order realizations, such leaves compose while preserving covariance. The resulting law is generally non-Gaussian and not determined by the composite kernel alone: distinct trees can share the same kernel yet differ in higher-order moments (Appendix C.3).

The resulting compiler uses a hierarchy of increasingly general execution rules: exact spectral compilation for stationary subtrees, feature-space compilation for finite-rank subtrees, Gaussian modulation for compatible mixed products, and covariance-preserving realization composition as fallback.

SPECTOR Generation Pipeline. The preceding results enable covariance-free compilation and sampling of compositional priors. We now describe the full SPECTOR pipeline, combining an expanded primitive bank with augmentations that add structure beyond kernel composition.

Expanded Kernel Bank demonstrates the compiler's extensibility: new primitives require only a realization, while the existing compilation rules apply unchanged. We expand KernelSynth's five primitives to 18, spanning stationary correlation, periodicity, noise, trend and growth, and regime/transient behavior (Table 2). The bank uses heterogeneous realizations: stationary kernels compile spectrally, finite-rank nonstationary kernels in feature space, and recursive kernels such as Wiener and integrated Wiener directly in time. This broadens the compositional vocabulary while remaining covariance-free and supporting non-Gaussian processes, such as Student-t noise with white-noise covariance and heavy-tailed marginals. SPECTOR samples $K \sim \mathrm{Uniform}\{1, \ldots, 5\}$ primitives from the weighted bank and folds them left-associatively into a sum–product program. Full definitions, hyperparameters, sampling probabilities, and realization details appear in Appendix F.1.

Augmentations. We complement the compositional prior with augmentations from recent synthetic-pretraining pipelines: SCM propagation through nonlinear DAGs for cross-series dependence (Xie et al., 2026), value-space transforms such as quantization (Moroshan et al., 2025), and TiRex-style spike injection and piecewise-linear amplitude modulation (Auer et al., 2026). These add nonlinear coupling, discrete observations, impulsive events, and time-varying scale beyond the base stochastic-process prior. Implementation details and parameter distributions are given in Appendix F.4.

Path Generation. Given a sampled sum–product tree, SPECTOR recursively dispatches each subtree to its strongest realization: spectral for stationary subtrees, feature-space for finite-rank subtrees, and direct time-domain sampling for structured nonstationary primitives. Mixed products use Gaussian modulation when applicable and otherwise the covariance-preserving composition of Proposition 2.1. Paths are evaluated directly on the requested grid and passed to the augmentation layer without materializing dense covariance. Full sampling algorithms appear in Appendix A.

3 Empirical Results

This section is structured as follows. First, we discuss the details of the benchmarks, baseline and the evaluation protocol. Then, we summarize key empirical results of SPECTOR in comparison to baseline synthetic priors and hybrid synthetic-real priors. Finally, we ablate the different design choices present in SPECTOR.

3.1 Benchmarks, baselines, models and training methodology

Benchmarks. We evaluate on three standard and comprehensive zero-shot forecasting benchmarks spanning diverse domains, frequencies, and horizon lengths. (1) **M-Series** (Makridakis et al., 1982; Makridakis & Hibon, 2000; Makridakis et al., 2020; Athanasopoulos et al., 2011) totaling over

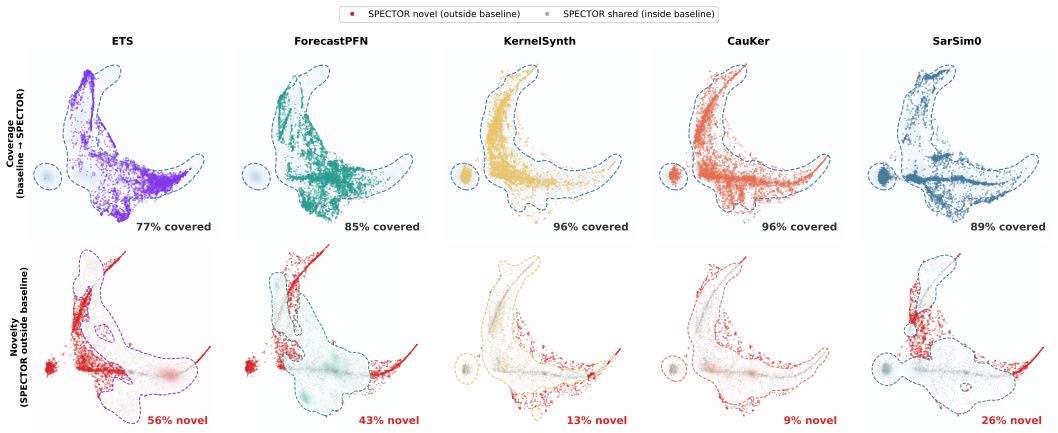


Figure 2: **SPECTOR covers 77-96% of prior generators while introducing additional coverage.** UMAP density contours of SPECTOR vs. competing synthetic priors. **Top row**: baseline points overlaid on SPECTOR's 95%-KDE contour (coverage). **Bottom row**: SPECTOR points outside the baseline's contour highlighted in red (novelty).

100,000 series at yearly, quarterly, monthly, and weekly frequencies. (2) **GiftEval** (Aksu et al., 2024) provides 97 univariate and multivariate tasks across seven domains, ten frequencies and evaluated over short, medium, and long horizons. (3) **FEV-Bench** (Shchur et al., 2025) covers frequencies from subhourly to yearly across univariate, multivariate, and covariate tracks. Since we only focus on univariate forecasting, all tasks are treated as univariate forecasting problems in which multivariate series are converted to univariate and covariates are unused. We report Skill Score (↑) and Pairwise Win Rate Shchur et al. (2025). The underlying metric is scaled CRPS (sCRPS) for M-Series and GiftEval, and scaled quantile loss (SQL) for FEV-Bench (more details are in Appendix H; point-forecasting results using MASE are reported in Appendix I.

Baselines. We compare SPECTOR against five synthetic data generators spanning kernel-based and parametric simulation paradigms. **KernelSynth** (Ansari et al., 2024) samples Gaussian processes from random sums and products of five primitive kernels: RBF, ExpSineSquared, RationalQuadratic, linear, and white noise. **ForecastPFN** (Dooley et al., 2023) generates trend-seasonality-noise trajectories from a multiplicative structural model with Weibull-distributed noise. **ETS** (Hyndman et al., 2008) simulates the error-trend-seasonality state-space family, including damped trends and multiplicative errors. **SarSim0** (Oreshkin et al., 2026) combines seasonal ARIMA trajectories with bi-seasonal superposition and heavy-tailed noise, while **CauKer** (Xie et al., 2026) augments KernelSynth with randomized structural causal model to induce non-linear transformations and dependencies across generated series.

Models and training methodology. To evaluate the model-agnostic effectiveness of all synthetic generators including SPECTOR, we consider six variants spanning wide architectural spectrum: the MLP-based N-BEATS (Oreshkin et al., 2020), the CNN-based MQCNN (Wen et al., 2017), Transformer-based PatchTST-FM (Wen et al., 2026) and Chronos2 (Ansari et al., 2025), the state-space-based S4D (Gu et al., 2022; Yu et al., 2025), and the xLSTM-based TiRex (Auer et al., 2026). PatchTST-FM, Chronos2 and TiRex use patching; the remaining models process the sequence directly. All models are trained for probabilistic forecasting using the multi-quantile pinball loss (Koenker & Bassett, 1978). We use a shared training protocol across pretraining generators. Training uses a maximum sequence length $L = 1024$ and forecast horizon $H = 128$. Model-specific architectural and optimization details are provided in Appendix G.

3.2 Main Experimental Results

We evaluate SPECTOR along five axes. (i) **Coverage:** Does SPECTOR cover a broader set of temporal patterns when compared to existing synthetic generators? (ii) **Architectural Robustness:** Are the benefits of SPECTOR architecture agnostic? (iii) **Necessity:** Can SPECTOR be recovered by a mixture of prior generators? (iv) **Scalability:** Is SPECTOR efficient enough for on-the-fly generation at the context lengths used by modern TSFMs? (v) **Competitiveness:** Can SPECTOR pretrain TSFMs that are competitive with analogs trained on real and synthetic corpora?

Table 1: **Pretraining with SPECTOR positively transfers across benchmarks and model architectures.** Zero-shot Probabilistic Forecasting Performance at context length of 1024 (Skill Score % ↑). Skill Scores are computed w.r.t sCRPS for M-Series and GiftEval, and w.r.t SQL for FEV-Bench. SPECTOR consistently outperforms all baseline generators across all models and benchmarks. **Bold** indicates best and underlined indicates second best.

Generator	N-BEATS	MQCNN	Chronos-2	PatchTST-FM	S4D	TiRex
M-Series						
ETS	13.20	7.65	8.83	12.76	12.97	9.46
ForecastPFN	5.47	1.80	6.20	9.51	7.57	9.91
KernelSynth	1.45	1.55	−0.44	1.94	6.07	−1.51
CauKer	9.21	11.49	7.10	13.23	15.41	12.35
SarSim0	13.30	1.37	15.51	14.88	17.38	13.65
SPECTOR (Ours)	**17.72**	**16.84**	**16.20**	**20.96**	**21.21**	**18.02**
GiftEval						
ETS	15.36	10.02	22.40	19.75	22.34	15.55
ForecastPFN	20.56	9.44	23.03	17.21	19.33	12.74
KernelSynth	29.21	26.65	30.73	30.00	32.14	23.41
CauKer	36.64	33.99	41.23	39.76	40.90	38.95
SarSim0	30.25	23.13	31.00	32.29	32.11	21.55
SPECTOR (Ours)	**40.33**	**35.28**	**45.56**	**43.72**	**43.03**	**41.30**
FEV-Bench						
ETS	24.37	20.33	23.79	25.16	26.02	22.52
ForecastPFN	26.28	19.29	26.75	28.78	27.91	25.19
KernelSynth	25.85	23.21	24.93	28.50	30.53	22.52
CauKer	34.28	31.80	33.78	36.59	37.99	34.98
SarSim0	29.62	22.94	31.75	31.34	32.19	29.26
SPECTOR (Ours)	**36.58**	**34.08**	**37.63**	**39.99**	**39.96**	**37.43**

Coverage. To study the coverage of SPECTOR we generate 5,000 length-1024 series from each generator, and embed all max-normalized series jointly into a two-dimensional UMAP space. For each source, we fit a kernel density estimate (KDE) and define its 95%-density contour. We measure *coverage* of a baseline by SPECTOR as the fraction of baseline samples falling inside the SPECTOR contour, and *novelty* as the fraction of SPECTOR samples falling outside the corresponding baseline contour. Figure 2 reveals two consistent patterns. (i) Baseline samples largely lie within the region occupied by SPECTOR, with 77–96% of each baseline covered by its contour showing that SPECTOR captures most temporal patterns. (ii) Conversely, 9–56% of SPECTOR samples fall outside the corresponding baseline contours, indicating that its expanded kernel bank and composition rules generate substantial structure not represented by any single competing prior. Together, these results suggest that SPECTOR provides a broader empirical pretraining distribution: it captures much of the variation produced by existing generators while extending into additional regions of the time-series representation space.

Architectural Robustness. Table 1 shows how SPECTOR achieves the best average Skill Score in every one of the 18 architecture-benchmark combinations, outperforming ETS, ForecastPFN, KernelSynth, CauKer, and SarSim0 across all three benchmarks. The gains persist across substantially different benchmarks, indicating that the compositional prior provides a broadly transferable training signal. To validate that the performance gain of SPECTOR is robust to benchmark composition, we also report SPECTOR's pairwise win rate with 95% confidence interval in Figure 1(b), demonstrating that SPECTOR outperforms baseline generators 74–91% across all tasks in the three benchmarks. The advantage of SPECTOR holds across MLP, CNN, Transformer, state-space, and xLSTM architectures, suggesting that the gains arise primarily from the SPECTOR prior diversity rather than from model family compatibility. Additional metrics and frequency / horizon breakdowns are reported in Appendix I.

Necessity. We now examine how the performance of SPECTOR changes when mixed with other synthetic generators as recent work suggests that mixtures outperform single source training (Nagpal et al., 2026; Moroshan et al., 2025). To this end, we train PatchTST-FM on a uniform mixture of all six generators and also on a uniform mixture that includes all but one source. As seen in figure 6 in Appendix I.7, the full mixture improves over some single sources like SarSim0 and CauKer but not over SPECTOR on M-Series and FEV-Bench while marginally (+0.2) on GiftEval. Moreover, removing SPECTOR causes the largest degradation among mixture ablations, reducing Skill Score

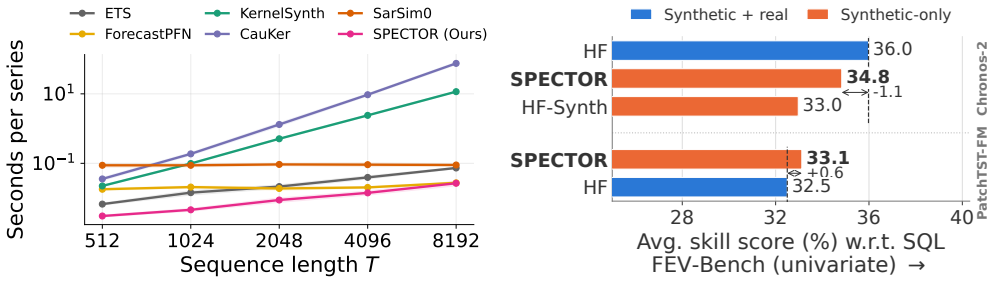


Figure 3: **SPECTOR scales better that other generators while being competitive with checkpoints trained on real and synthetic data. Left:** Generation time per series (single CPU core, mean over 10 runs) vs. sequence length T. **Right:** SPECTOR-pretrained models against pretrained checkpoints approach or exceed real-synthetic pretrained checkpoints in Fev-Bench univariate.

by 2.5 points on GiftEval and 3.0 points on FEV-Bench. These results suggest that the evaluated generator mixture does not recover SPECTOR's transfer benefit.

Scalability. We benchmark all synthetic generators on a single CPU core using batches of 64 series with T up to 8192, reporting the mean over 10 runs; SPECTOR uses $M = 64$ spectral features. Figure 3 (left) shows two distinct scaling regimes. While KernelSynth and CauKer scale cubically with sequence length due to dense covariance factorization, SPECTOR *scales linearly with* T, closely tracking fast parametric simulators such as ETS while retaining compositional kernel diversity. At $T = 8192$, SPECTOR remains subseconds per series, making on-the-fly generation during training practical even at long contexts.

Competitiveness. A key question for synthetic-only pretraining is how closely it can approach models pretrained on large real-world corpora. We compare SPECTOR-pretrained Chronos-2 (Ansari et al., 2025) and PatchTST-FM (Wen et al., 2026) against their published HuggingFace checkpoints pretrained on real and synthetic data. Both SPECTOR-pretrained variants are trained with maximum context length of 8192, matching the HF variants. Because Chronos-2 pretraining data overlaps with GiftEval and our variants are pretrained on univariate forecasting only, all variants are evaluated using univariate inference mode on the Fev-Bench univariate subset. For Chronos-2, we also include the published synthetic-only checkpoint (HF-synth), pretrained on Chronos-2's own mixture of synthetic generators. Figure 3 (right) shows two findings. (i) *Near-parity with hybrid pretraining:* PatchTST-FM pretrained solely on SPECTOR exceeds its real+synthetic HF checkpoint (33.1 vs. 32.5 Skill Score), while Chronos-2 trails its corresponding HF checkpoint by only 1.2 points. (ii) *Stronger synthetic-only pretraining:* for Chronos-2, SPECTOR also outperforms the published synthetic-only checkpoint by 1.9 points (34.8 vs. 33.0), despite using a single synthetic prior rather than a mixture of generators. These results highlight the breath of patterns captured by SPECTOR which do not necessitate real-world corpora to achieve the best results.

3.3 Ablation Studies

We conduct a series of ablations using PatchTST-FM to isolate the impact of the SPECTOR sampling procedure, augmentation, kernel-bank diversity, and sequence length (Figure 4 and Table 26).

Finite-feature realization turns approximation into a pretraining advantage. Holding the kernel bank fixed to KernelSynth's legacy five-kernel prior, with no augmentation and the same pretraining sample budget for PatchTST-FM, Figure 4 shows that SPECTOR's finite-feature realization improves downstream performance by +2.5–6.3 pp across three benchmarks. Although the compiled spectral law is exact, finite-M path synthesis is not an exact GP draw; we hypothesize that this induces *spectral regularization*, constraining the effective spectral complexity of composite kernels through a finite set of sampled frequencies. This effect becomes increasingly pronounced for deeper compositions, where we observe up to $1.5\times$ lower mid- and high-frequency energy while retaining the low-frequency structure of the exact prior (Figure 8). This spectral attenuation provides a possible connection to the role of high-frequency suppression in *diffusability* (Skorokhodov et al., 2025), and might explain why the finite-feature approximation can improve rather than degrade pretraining performance. We provide further observational analysis of the differences between the two sampling procedures in Appendix J, while leaving a causal analysis of this phenomenon to future work.

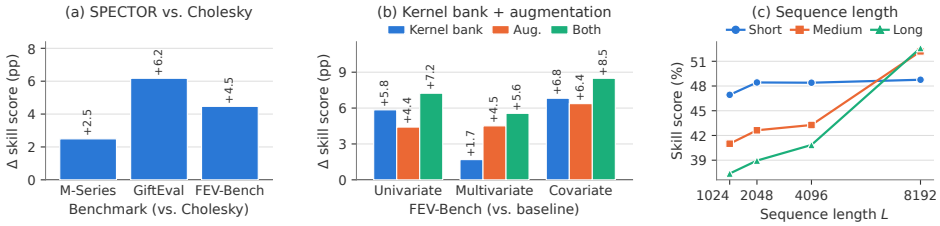


Figure 4: **Ablation results** using PatchTST-FM on the effect of (a) SPECTOR sampling, (b) kernel-bank expansion and augmentation, and (c) sequence length. Full results are reported in Table 26.

Expanded kernel bank and augmentation provide complementary benefits. We next study the role of expanded kernel bank and augmentation within SPECTOR. Figure 4 shows that expanding the kernel bank yields significant gains on univariate and covariate subsets (univariate with unused covariates) of FEV-Bench, suggesting that increased kernel diversity primarily benefits the diversity of univariate temporal dynamics. Adding augmentations improves the performance across all subsets but shows better advantage than kernel bank expansion in the multivariate subset. Combining kernel bank expansion and augmentation produces the strongest overall performance, indicating that the two components capture complementary sources of diversity and structure.

Longer sequences improve forecasting, motivating scalable sampling. Increasing the pretraining sequence length from 1024 to 8192 substantially improves performance, especially at longer horizons, up to +15.23 pp on GiftEval long horizons. These gains motivate a sampler that scales linearly with sequence length for computationally feasible long-context synthetic pretraining.

SPECTOR remains the best synthetic prior even under shared augmentation. To further isolate the contribution of the generator, we apply the same SPECTOR augmentation to baseline generators. SPECTOR outperforms all augmented baselines across all three benchmarks by +1.89/+1.75/+1.91 pp on M-Series, GiftEval, and FEV-Bench (Table 24 in Appendix I.6). To validate that the improvements are robust to benchmark compositions, we follow Shchur et al. (2025) and report task-level pairwise win rates of SPECTOR against the two strongest augmented baselines, with 95% confidence intervals obtained by a paired bootstrap over the 210 tasks (Table 23). SPECTOR wins on 70.0% and 71.9% of tasks against KernelSynth and CauKer, respectively, with both 95% bootstrap CIs well above 50%.

4 Conclusion

We introduced SPECTOR, a realization compiler that makes compositional time-series priors directly executable. By preserving the structure of the kernel sum–product programs and compiling them into efficient realization spaces, SPECTOR avoids dense covariance sampling while retaining broad stochastic-process diversity. This yields linear-in-length generators suitable for long-context, on-the-fly pretraining. Across six forecasting architectures and three benchmark suites, SPECTOR consistently outperforms competing synthetic generators, showing that computational efficiency need not come at the expense of prior diversity or downstream transfer. More broadly, SPECTOR makes the pretraining distribution itself programmable: enabling *in-situ pretraining* with effectively unbounded synthetic data generated on demand; *data-induced capability design* through controlled interventions on the generative prior; and, prospectively, *agentic prior optimization*, where the generator is iteratively modified and evaluated in an automated loop.

Limitations and future work. SPECTOR leaves a few important directions open. First, generation is empirically fast, but linear-in-T complexity assumes bounded composition size. Deep finite-rank products can multiply feature dimension, and modulation costs scale with rank. Second, realization classes induce different laws beyond covariance. Finite-M spectral synthesis departs from the GP limit yet improves pretraining, suggesting a potentially useful finite-feature regularization effect that remains theoretically unexplained; more broadly, covariance-consistent path composition admits non-Gaussian laws whose potential remains unexplored. Third, the compiler currently uses spectral, feature-space, and direct realizations; other compatible representation algebras, such as wavelets, may unlock additional efficient prior classes. Fourth, the pretraining prior remains manually specified through its primitives, grammar, hyperparameters, and augmentations; learning or adapting this programmable prior automatically remains open.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs, to implement methods, to interpret results, to provide critical ingredients for proving mathematical claims, to formulate mathematical claims, and to provide feedback on our research methodology and experimental design. We have not used generative AI tools to generate synthetic data sets, to develop theoretical models or conceptual frameworks, to propose or refine hypotheses, or to clean or reformat datasets; assistance with translation and support for qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create and modify scientific figures, to draft parts of this manuscript, to summarize and analyze existing literature, and to source and search for information.

We have reviewed all AI-assisted work. All AI-assisted code was read, verified, and tested for correctness by the authors, and every experimental result reported in this paper was produced by code the authors have inspected. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

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

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

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.

George Athanasopoulos, Rob J. Hyndman, Haiyan Song, and Doris C. Wu. The tourism forecasting competition. *International Journal of Forecasting*, 27(3):822–844, 2011.

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. *Advances in Neural Information Processing Systems*, 38:57529–57580, 2026.

Andrew C. Bauer, Hasan Abbasi, James Ahrens, Hank Childs, Berk Geveci, Scott Klasky, Kenneth Moreland, Patrick O'Leary, Venkatram Vishwanath, Brad Whitlock, and E. Wes Bethel. In situ methods, infrastructures, and applications on high performance computing platforms. *Computer Graphics Forum*, 35(3):577–597, 2016. doi: 10.1111/cgf.12930.

Vidmantas Bentkus. On the dependence of the Berry–Esseen bound on dimension. *Journal of Statistical Planning and Inference*, 113(2):385–402, 2003.

Alain Berlinet and Christine Thomas-Agnan. Reproducing Kernel Hilbert Spaces in Probability and Statistics. *Springer*, 2012.

Salomon Bochner. Monotone funktionen, stieltjessche integrale und harmonische analyse. *Mathematische Annalen*, 108:378–410, 1933.

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha Naidu, and Colin White. ForecastPFN: Synthetically-trained zero-shot forecasting. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2311.01933.

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1166–1174, 2013.

Jacob R. Gardner, Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. GPyTorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Jonathan S. Golan. *Semirings and their Applications*. Kluwer Academic Publishers, Dordrecht, 1999. doi: 10.1007/978-94-015-9333-5.

Ulf Grenander, Gábor Szegö, and Griffith C. Evans. Toeplitz Forms and Their Applications. *Literary Licensing, LLC*, 2013.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022.

Rob Hyndman, Anne Koehler, Keith Ord, and Ralph Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer, 2008.

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

Roger Koenker and Gilbert Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978.

James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

Spyros Makridakis and Michèle Hibon. The M3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000.

Spyros Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, 1(2):111–153, 1982.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1): 54–74, 2020.

Vladyslav Moroshan, Julien Siems, Arber Zela, Timur Carstensen, and Frank Hutter. TempoPFN: Synthetic pre-training of linear rnns for zero-shot time series forecasting, 2025. URL https://arxiv.org/abs/2510.25502.

Riley Murray, Jim Demmel, Michael W. Mahoney, N. Benjamin Erichson, Maksim Melnichenko, Osman Asif Malik, Laura Grigori, Piotr Luszczek, Michał Dereziński, Miles E. Lopes, Tianyu Liang, Hengrui Luo, and Jack Dongarra. Randomized numerical linear algebra: A perspective on the field with an eye to software, 2023. URL https://arxiv.org/abs/2302.11474.

Aaryan Nagpal, Debdeep Sanyal, Murari Mandal, Dhruv Kumar, and Saurabh Deshpande. Mix, don't pick: Why synthetic corpus composition matters for time series foundation model pretraining. *arXiv preprint arXiv:2606.09912*, 2026.

Kin G. Olivares, Malcolm Wolff, Tatiana Konstantinova, Shankar Ramasubramanian, Boris N. Oreshkin, Andrew Gordon Wilson, Andres Potapczynski, Willa Potosnak, Michael W. Mahoney, Mengfei Cao, and Dmitry Efimov. A more realistic evaluation of cross-frequency transfer learning and foundation forecasting models. In *NeurIPS 2025 Workshop on Recent Advances in Time Series Foundation Models: Have We Reached the "BERT Moment"?*, 2025.

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. URL https://openreview.net/forum?id=r1ecqn4YwB.

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

Carl Edward Rasmussen and Christopher K. I. Williams. Gaussian Processes for Machine Learning. *MIT Press*, 2006.

Walter Rudin. Fourier Analysis on Groups. *Dover Publications*, 2017.

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.

Ivan Skorokhodov, Sharath Girish, Benran Hu, Willi Menapace, Yanyu Li, Rameen Abdal, Sergey Tulyakov, and Aliaksandr Siarohin. Improving the diffusability of autoencoders. *arXiv preprint arXiv:2502.14831*, 2025.

Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

Yunshi Wen, Wesley M Gifford, Chandra Reddy, Lam M Nguyen, Jayant Kalagnanam, and Anak Agung Julius. Revisiting the generic transformer: Deconstructing a strong baseline for time series foundation models. *arXiv preprint arXiv:2602.06909*, 2026.

Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1775–1784. PMLR, 2015.

Shifeng Xie, Vasilii Feofanov, Marius Alonso, Ambroise Odonnat, Jianfeng Zhang, Themis Palpanas, and Ievgen Redko. CauKer: Classification time series foundation models can be pretrained on synthetic data only. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2508.02879.

Annan Yu, Dongwei Lyu, Soon Hoe Lim, Michael W. Mahoney, and N. Benjamin Erichson. Tuning frequency bias of state space models. In *International Conference on Learning Representations*, 2025.

Annan Yu, Danielle Maddix, Boran Han, Xiyuan Zhang, Abdul Fatir Ansari, Oleksandr Shchur, Christos Faloutsos, Andrew Gordon Wilson, Michael W Mahoney, and Bernie Wang. Understanding the implicit biases of design choices for time series foundation models. In *International Conference on Learning Representations*, volume 2026, pp. 71998–72049, 2026.

A COMPLETE SPECTOR GENERATION PROCEDURE.

We decompose the implementation into three stages. Algorithm 1 samples a compositional prior from the weighted primitive bank. Algorithm 2 shows full SPECTOR generation pipeline. Algorithm 3 compiles and realizes the sampled expression using the strongest applicable realization operator. Stationary subtrees invoke the spectral-tree sampler of Algorithm 4.

Algorithm 1 Sampling a compositional prior

Require: Weighted primitive bank $\mathcal{B}$
Ensure: Expression tree $\mathcal{T}$
Draw $K \sim \mathrm{Uniform}\,1, 2, 3, 4, 5$
for $j = 1, \ldots, K$ **do**
    Sample primitive k_j from $\mathcal{B}$ according to its weight
    Sample the hyperparameters of k_j
end for
$\mathcal{T} \leftarrow k_1$
for $j = 2, \ldots, K$ **do**
    Draw $u \sim \mathrm{Bernoulli}(0.5)$
    if $u = 1$ **then**
        $\mathcal{T} \leftarrow \mathrm{Sum}(\mathcal{T}, k_j)$
    else
        $\mathcal{T} \leftarrow \mathrm{Prod}(\mathcal{T}, k_j)$
    end if
end for
return $\mathcal{T}$

Algorithm 2 Full SPECTOR generation pipeline

Require: Time points $t_1, \ldots, t_T$, primitive bank $\mathcal{B}$, spectral feature count M
Require: $N_{\text{nodes}} = 8$, $N_{\text{channels}} = 4$
Ensure: Augmented synthetic paths
function GENERATEBASEPATH($t_{1:T}, M, \mathcal{B}$)
    $\mathcal{T} \leftarrow$ SAMPLEEXPRESSIONTREE($\mathcal{B}$)
    return REALIZE($\mathcal{T}, t_{1:T}, M$)
end function
Sample a DAG G with $N_{\text{nodes}} = 8$ and maximum in-degree 6
for each root node i of G **do**
    $z_i \leftarrow$ GENERATEBASEPATH($t_{1:T}, M, \mathcal{B}$)
end for
for each non-root node i in topological order **do**
    Sample $\sigma_i \in \mathrm{linear}, \mathrm{ReLU}, \mathrm{sigmoid}, \sin, \mathrm{mod}, \mathrm{leakyReLU}$
    Sample $w_{ij} \sim \mathcal{N}(0, 1)$ for $j \in \mathrm{pa}(i)$
    $z_i \leftarrow \sigma_i \left(\sum_{j \in \mathrm{pa}(i)} w_{ij} z_j + b_i \right)$
end for
Select $N_{\text{channels}} = 4$ output paths Y from G
for each path y in Y **do**
    Draw $u \sim \mathrm{Bernoulli}(0.5)$
    if $u = 1$ **then**
        Select uniformly $A \in \mathrm{Quantization}, \mathrm{SpikeInjection}, \mathrm{AmplitudeModulation}$
        Sample the parameters of A
        $y \leftarrow A(y)$
    end if
end for
return Y

Algorithm 3 SPECTOR realization compiler

Require: Expression node v, time points $t_1, \dots, t_T$, spectral feature count M
Ensure: Realized path $f(t_{1:T})$

function REALIZE($v, t_{1:T}, M$)
  if v is a stationary subtree **then**
    return SPECTRALTREESAMPLE($v, t_{1:T}, M$) ▷ Algorithm 4
  end if
  if v is a finite-rank subtree **then**
    $\psi_v \leftarrow$ COMPILEFEATUREMAP(v) ▷ $+ \mapsto \oplus,\ \times \mapsto \otimes$
    Draw $z \sim \mathcal{N}(0, I_{\dim(\psi_v)})$
    return $\big(\psi_v(t_1)^\top z, \dots, \psi_v(t_T)^\top z\big)$
  end if
  if v is a directly realizable leaf **then**
    return DIRECTSAMPLE($v, t_{1:T}$)
  end if
  if $v = \mathrm{Sum}(a, b)$ **then**
    $f_a \leftarrow$ REALIZE($a, t_{1:T}, M$)
    $f_b \leftarrow$ REALIZE($b, t_{1:T}, M$) independently
    return $f_a + f_b$
  end if
  if $v = \mathrm{Prod}(a, b)$ **then**
    if a is finite rank and b is Gaussian-realizable **then**
      $\psi_a = (\psi_{a,1}, \dots, \psi_{a,r}) \leftarrow$ COMPILEFEATUREMAP(a)
      for $j = 1, \dots, r$ **do**
        $g_j \leftarrow$ REALIZE($b, t_{1:T}, M$) ▷ independent across j
      end for
      return $\displaystyle\sum_{j=1}^{r} \psi_{a,j}(t_{1:T}) \odot g_j$
    end if
    if b is finite rank and a is Gaussian-realizable **then**
      $\psi_b = (\psi_{b,1}, \dots, \psi_{b,r}) \leftarrow$ COMPILEFEATUREMAP(b)
      for $j = 1, \dots, r$ **do**
        $g_j \leftarrow$ REALIZE($a, t_{1:T}, M$) ▷ independent across j
      end for
      return $\displaystyle\sum_{j=1}^{r} \psi_{b,j}(t_{1:T}) \odot g_j$
    end if
    $f_a \leftarrow$ REALIZE($a, t_{1:T}, M$)
    $f_b \leftarrow$ REALIZE($b, t_{1:T}, M$) independently
    return $f_a \odot f_b$ ▷ covariance-preserving fallback
  end if
end function

A.1 SPECTRAL-TREE SAMPLER.

Algorithm 4 gives a direct implementation of Theorem 2.3 and the subsequent finite-feature realization in Proposition D.2. Each leaf kernel k_ℓ is assumed to provide (i) its zero-lag mass $k_\ell(0)$ and (ii) a sampler from its normalized spectral law $p_\ell(d\omega) = d\mu_\ell(\omega)/k_\ell(0)$. The zero-lag mass of every internal node is computed once by a bottom-up pass: sum nodes add child masses, while product nodes multiply them. Frequency draws can then be generated recursively without constructing the composite kernel, spectral density, or covariance matrix.

Algorithm 4 Spectral-tree sampling for a stationary compositional prior

Require: Expression tree $\mathcal{T}$ with stationary kernel leaves
Require: Leaf spectral samplers p_ℓ and masses $k_\ell(0)$
Require: Time points $t_1, \dots, t_T$ and number of spectral features M
Ensure: Sample path $(f_M(t_1), \dots, f_M(t_T))$

function MASS(v)
  if v is a leaf k_ℓ **then**
    return $k_\ell(0)$
  else if $v = \mathrm{Sum}(a, b)$ **then**
    return MASS(a) + MASS(b)
  else ▷ $v = \mathrm{Prod}(a, b)$
    return MASS(a) × MASS(b)
  end if
end function

function DRAWFREQUENCY(v)
  if v is a leaf k_ℓ **then**
    return $\omega \sim p_\ell$
  else if $v = \mathrm{Sum}(a, b)$ **then**
    $\pi_a \leftarrow \dfrac{\text{MASS}(a)}{\text{MASS}(a) + \text{MASS}(b)}$
    Draw $z \sim \mathrm{Bernoulli}(\pi_a)$
    if $z = 1$ **then**
      return DRAWFREQUENCY(a)
    else
      return DRAWFREQUENCY(b)
    end if
  else ▷ $v = \mathrm{Prod}(a, b)$
    $\omega_a \leftarrow$ DRAWFREQUENCY(a)
    $\omega_b \leftarrow$ DRAWFREQUENCY(b) independently
    return $\omega_a + \omega_b$
  end if
end function

Compute and cache MASS(v) for every node v in a bottom-up pass
for $m = 1, \dots, M$ **do**
  $\omega_m \leftarrow$ DRAWFREQUENCY($\mathcal{T}$)
  $b_m \sim \mathrm{Unif}[0, 2\pi]$
end for
$A \leftarrow \sqrt{2\,\text{MASS}(\mathcal{T})/M}$
for $i = 1, \dots, T$ **do**
  $f_M(t_i) \leftarrow A \sum_{m=1}^{M} \cos(\omega_m t_i + b_m)$
end for
return $(f_M(t_1), \dots, f_M(t_T))$

The recursive calls in DRAWFREQUENCY implement the normalized spectral law exactly: a sum node becomes a mixture weighted by the child spectral masses, while a product node becomes con-

volution through addition of independent frequency draws. The only finite-feature approximation enters after these exact draws are obtained, when M sampled frequencies are used to construct the time-domain realization. With cached node masses, drawing M frequencies costs $O(LM)$ in the worst case for a tree with L leaves, and evaluating the resulting expansion over T time points costs $O(MT)$.

B Related Work

Kernel-based generators define rich synthetic priors by composing primitive covariance functions through sums and products. KernelSynth (Ansari et al., 2024) has become a major synthetic source for TSFM pretraining, including Chronos and TiRex (Auer et al., 2026), while CauKer (Xie et al., 2026) extends KernelSynth with structural causal mechanisms for improved generalization. Recent evidence further supports this compositional family: across 11 synthetic generators and two TSFM architectures, KernelSynth delivers the strongest single-generator CRPS (Nagpal et al., 2026). The "catch" is that conventional realization collapses this rich prior back into a dense covariance matrix, with $O(T^2)$ memory and $O(T^3)$ factorization. Structured, low-rank, randomized, and GPU-accelerated methods can reduce this cost (Wilson & Nickisch, 2015; Gardner et al., 2018; Xie et al., 2026; Murray et al., 2023), but they still route sampling through covariance computation, leaving long-context in-situ generation elusive. Our approach makes kernel compositions $O(T)$ by construction and supports both Gaussian and non-Gaussian realizations, including Student-t processes.

State-space and direct functional generators offer efficient sampling by committing to specific parametric constructions. ETS (Hyndman et al., 2008) evolves error, trend, and seasonal states recursively, while SarSim0 (Oreshkin et al., 2026) combines stable seasonal ARIMA dynamics with bi-seasonal superposition and heavy-tailed noise. ForecastPFN (Dooley et al., 2023) instead multiplicatively combines linear and exponential trends, Fourier seasonal components, and independent Weibull noise. These approaches enable on-the-fly generation, but they are limited by constrained simulator templates. In contrast, SPECTOR retains the open-ended compositional structure of kernel priors and achieves linear-time sampling by changing how those priors are realized, rather than by restricting the family of processes they can express.

Theoretical foundations for kernel realization. The main theoretical building blocks for our spectral semiring proposition come from various sources. For general positive-definite kernels, including nonstationary kernels, we rely on the Reproducing Kernel Hilbert Spaces (RKHS) and Mercer representation framework from Berlinet & Thomas-Agnan (2012), together with its connection to covariance kernels of stochastic processes. This framework also provides concrete examples of kernels beyond stationary ones that we use in our construction. In particular, for stationary kernels we use Bochner theorem introduced in Bochner (1933) and also treated in a general manner in Rudin (2017). Most of the kernels that we selected (particularly the stationary ones) are heavily inspired by the use of Gaussian Processes in ML as treated in Rasmussen & Williams (2006). Finally, for stationary kernels sampled on a regular grid, the resulting covariance kernel matrices have Toeplitz structure, connecting the corresponding continuous kernel convolution operators with their finite-dimensional discretizations; this relationship is treated in Grenander et al. (2013).

C Compositional Kernel Programs: Examples and Technical Details

This section expands the algebraic viewpoint introduced in Section 2. We distinguish between the *syntax* of a compositional kernel program and its *semantics*. The syntax is an expression tree built from primitive kernels using sum and product operators; the semantics is the positive-semidefinite kernel denoted by evaluating that tree in the kernel semiring.

Syntax and kernel-space evaluation. Let $\mathfrak{T}(\mathcal{B})$ denote the smallest set of expressions satisfying the following recursive definition:

$$k \in \mathcal{B} \quad \Longrightarrow \quad k \in \mathfrak{T}(\mathcal{B}),$$

and whenever $\mathcal{T}_1, \mathcal{T}_2 \in \mathfrak{T}(\mathcal{B})$,

$$\mathrm{Sum}(\mathcal{T}_1, \mathcal{T}_2) \in \mathfrak{T}(\mathcal{B}), \qquad \mathrm{Prod}(\mathcal{T}_1, \mathcal{T}_2) \in \mathfrak{T}(\mathcal{B}).$$

Thus, a compositional kernel program is either a primitive kernel from $\mathcal{B}$ or a finite expression obtained by recursively combining such kernels through sums and products. Importantly, an expression tree is a syntactic program, whereas $\mathrm{Eval}_K(\mathcal{T})$ is the kernel represented by that program. Distinct trees may therefore represent the same kernel. For example,

$$(k_1 + k_2)k_3 \qquad \text{and} \qquad k_1k_3 + k_2k_3$$

are different expression trees but have identical kernel-space evaluations.

Proposition C.1 (Positive-semidefinite kernel semiring)**.** *Let $\mathcal{K}_+$ denote the set of positive-semidefinite kernels on $\mathcal{X}$. Under kernel addition*

$$(k_1 + k_2)(x, x') = k_1(x, x') + k_2(x, x')$$

and pointwise multiplication

$$(k_1k_2)(x, x') = k_1(x, x')k_2(x, x'),$$

$\mathcal{K}_+$ forms a commutative semiring.

Proof. Closure under addition follows directly from positive semidefiniteness. For any finite set $x_1, \ldots, x_n$ and coefficients $c_1, \ldots, c_n$,

$$\sum_{i,j} c_ic_j(k_1 + k_2)(x_i, x_j) = \sum_{i,j} c_ic_jk_1(x_i, x_j) + \sum_{i,j} c_ic_jk_2(x_i, x_j) \geq 0. \tag{2}$$

For multiplication, let K_1 and K_2 denote the Gram matrices of k_1 and k_2 on an arbitrary finite input set. The Gram matrix associated with k_1k_2 is

$$K_1 \odot K_2,$$

where $\odot$ denotes the Hadamard product. By the Schur product theorem, $K_1 \odot K_2$ is positive semidefinite, establishing closure under pointwise multiplication.

Associativity and commutativity follow from the corresponding pointwise properties of real addition and multiplication. Distributivity also holds pointwise:

$$k_1(k_2 + k_3) = k_1k_2 + k_1k_3.$$

The zero kernel is the additive identity and the constant-one kernel is the multiplicative identity. Hence $\mathcal{K}_+$ is a commutative semiring. □

A concrete compositional program. Consider

$$\mathcal{T} = \mathrm{Prod}(\mathrm{Sum}(k_1, k_2), k_3)\,.$$

In the corresponding expression tree, the primitive kernels k_1, k_2, k_3 form the *leaves*, while sum and product operations form the internal nodes. Kernel-space evaluation gives

$$\mathrm{Eval}_K(\mathcal{T})(x, x') = \big(k_1(x, x') + k_2(x, x')\big)k_3(x, x').$$

For example, k_1 might represent smooth variation, k_2 periodic variation, and k_3 a nonstationary trend component. Their composition therefore defines a process whose first two structures are modulated by the third.

If, as before, K_j denotes the Gram matrix of k_j on a finite input grid $x_1, \ldots, x_n$, then:

$$K_{\mathcal{T}} = (K_1 + K_2) \odot K_3.$$

Distributivity gives

$$(K_1 + K_2) \odot K_3 = (K_1 \odot K_3) + (K_2 \odot K_3),$$

providing a finite-dimensional view of the same semiring structure. Thus, kernel addition corresponds to covariance-matrix addition, while kernel multiplication corresponds to Hadamard multiplication.

The semiring generated by the primitive bank. The image

$$\mathrm{Eval}_K\big(\mathfrak{T}(\mathcal{B})\big) \subseteq \mathcal{K}_+$$

is the subsemiring generated by the primitive bank $\mathcal{B}$. The syntax contains more information than the resulting kernel because multiple expression trees may evaluate to the same semiring element. Equivalently, one may identify expression trees that have the same kernel-space evaluation; the corresponding equivalence classes recover the generated kernel subsemiring. This distinction underlies the compiler viewpoint used in the main text: $\mathcal{T}$ specifies the compositional program, while different evaluation maps execute the same program in different representation spaces.

Singular positive-semidefinite kernels. Positive semidefiniteness does not require invertibility. In particular, Gram matrices associated with kernels in $\mathcal{K}_+$ may contain zero eigenvalues. This does not affect the semiring construction: both matrix addition and Hadamard multiplication remain closed over positive-semidefinite matrices, including singular ones. No quotienting or regularization is therefore needed merely because a covariance matrix is singular. Nonsingularity is required only by later results that explicitly invoke an inverse or inverse square root of a covariance matrix. For example, the finite-dimensional Berry–Esseen argument in Corollary D.2 assumes a nonsingular covariance matrix in order to whiten the random vector.

Zero and constant kernels. The zero kernel

$$k_0(x, x') = 0$$

and constant-one kernel

$$k_{\mathbf{1}}(x, x') = 1$$

are valid positive-semidefinite kernels and provide the additive and multiplicative identities of the ambient semiring. They need not appear as sampled primitives in $\mathcal{B}$.

The zero kernel requires minor care in normalized spectral representations: because $k_0(0) = 0$, its spectral measure cannot be normalized into a probability law by division by $k_0(0)$. This is a degeneracy of the normalization, not of the kernel algebra; the zero kernel can simply be handled as the deterministic zero process. All nonzero stationary leaves used by the spectral sampler satisfy $k(0) > 0$.

Scaled primitives. Nonnegative scaling preserves positive semidefiniteness:

$$k \in \mathcal{K}_+, \quad a \geq 0 \quad \Longrightarrow \quad ak \in \mathcal{K}_+.$$

In SPECTOR, amplitudes and other scale parameters are treated as part of the parameterized primitive kernel. Thus the sampled expression grammar needs only the sum and product operators above; explicit scalar multiplication does not require an additional composition rule.

C.1 Example of Direct Realization-Space Evaluation

Consider the compositional program

$$\mathcal{T} = \mathrm{Prod}(\mathrm{Sum}(k_1, k_2), k_3)\,,$$

whose kernel-space semantics is

$$\mathrm{Eval}_K(\mathcal{T}) = (k_1 + k_2)k_3.$$

A conventional covariance-based sampler would first evaluate this composite kernel on the time grid,

$$K_{\mathcal{T}} = (K_1 + K_2) \odot K_3,$$

and then sample from a distribution with covariance $K_{\mathcal{T}}$, typically requiring a matrix factorization.

The realization-tree construction instead assigns independent zero-mean realizations

$$f_1,\ f_2,\ f_3, \qquad \mathrm{Cov}(f_j) = k_j,$$

to the three leaves and evaluates the same syntax directly in realization space:

$$\mathrm{Eval}_R(\mathcal{T}) = (f_1 + f_2)f_3.$$

Its covariance is

$$\mathrm{Cov}\left((f_1(t)+f_2(t))f_3(t),(f_1(t')+f_2(t'))f_3(t')\right) = \big(k_1(t,t')+k_2(t,t')\big)k_3(t,t') \tag{3}$$

$$= \mathrm{Eval}_K(\mathcal{T})(t,t'). \tag{4}$$

Thus the composite covariance is obtained implicitly through path composition: the algorithm samples the leaf processes and combines their realizations according to the expression tree, without ever constructing

$$(K_1+K_2)\odot K_3$$

or factorizing the resulting composite covariance matrix.

C.2 Spectral Compilation as a Commuting Diagram

Proposition 2.2 can be summarized by the commuting diagram

$$\begin{array}{ccc} \mathcal{T} & \xrightarrow{\mathrm{Eval}_K} & k_{\mathcal{T}} \in \mathcal{K}_{\mathrm{stat}} \\ \| & & \downarrow \mathcal{S} \\ \mathcal{T} & \xrightarrow{\mathrm{Eval}_{\mathcal{S}}} & \mu_{\mathcal{T}} \in \mathcal{M}_+. \end{array}$$

Equivalently,

$$\mathcal{S}(\mathrm{Eval}_K(\mathcal{T})) = \mathrm{Eval}_{\mathcal{S}}(\mathcal{T}).$$

The upper path first evaluates the compositional program in kernel space and then transforms the resulting kernel spectrally. The lower path transforms the primitive kernels first and evaluates the same expression tree directly in spectral-measure space, where kernel sums become measure addition and kernel products become convolution. The two paths are exactly equivalent.

Here $\mathcal{K}_{\mathrm{stat}}$, $\mathcal{M}_+$, and $\mathcal{S}$ are as in Proposition 2.2. The compiler uses only the homomorphism property of $\mathcal{S}$, that is, $+ \mapsto +$ and $\cdot \mapsto *$; injectivity is not needed computationally, but guarantees that the spectral representation retains all information about the composite kernel.

This diagram captures the algebraic part of the compiler: the expression-tree syntax is unchanged, but its operators acquire different semantics in spectral space. Note that the computational gain arises only after coupling this representation with the recursive frequency sampler of Theorem 2.3 and the $O(MT)$ synthesis procedure of Corollary D.2.

C.3 Free-Semiring Interpretation

The compositional syntax introduced in Section 2 admits a more abstract algebraic interpretation. Let $\mathcal{B}$ denote the bank of primitive kernels. The expression trees $\mathfrak{T}(\mathcal{B})$ defined above form the *term algebra* over $\mathcal{B}$, in which no two distinct trees are identified. Quotienting by the commutative-semiring axioms and adjoining the additive and multiplicative identities yields the free commutative semiring $\mathbb{N}[\mathcal{B}]$ (Golan, 1999). Here, *free* is meant in the algebraic sense: the generators in $\mathcal{B}$ satisfy no relations beyond those required by the axioms of a commutative semiring. Equivalently, elements of $\mathbb{N}[\mathcal{B}]$ may be viewed as formal polynomials in the primitive-kernel symbols with coefficients in $\mathbb{N}$. At this level, these expressions are syntax rather than kernels themselves. In particular, $(k_1+k_2)k_3$ and $k_1k_3+k_2k_3$ are distinct trees in $\mathfrak{T}(\mathcal{B})$ but the *same* polynomial in $\mathbb{N}[\mathcal{B}]$.

The defining universal property is that, for any commutative semiring R, every assignment of the generators

$$\phi : \mathcal{B} \to R$$

extends uniquely to a semiring homomorphism

$$\widehat{\phi} : \mathbb{N}[\mathcal{B}] \to R.$$

Thus, once each primitive is assigned an interpretation in a target semiring, the meaning of every finite sum-product program is determined by the addition and multiplication operations of that target semiring. Kernel-space evaluation is one such interpretation:

$$\mathrm{Eval}_K : \mathbb{N}[\mathcal{B}] \to \mathcal{K}_+,$$

where $\mathcal{K}_+$ denotes the semiring of positive-semidefinite kernels under kernel addition and pointwise multiplication. Eval_K is not injective, because the primitives satisfy relations in $\mathcal{K}_+$ beyond the semiring axioms. For example, a product of two squared-exponential kernels is again a squared-exponential kernel,

$$k_{\mathrm{SE}}(\sigma_1, \ell_1)\, k_{\mathrm{SE}}(\sigma_2, \ell_2) = k_{\mathrm{SE}}(\sigma_1\sigma_2, \ell_3), \qquad \ell_3^{-2} = \ell_1^{-2} + \ell_2^{-2},$$

so the product tree and the corresponding single leaf are different elements of $\mathbb{N}[\mathcal{B}]$ with the same kernel semantics.

The spectral and finite-rank constructions developed in the main text provide alternative homomorphic interpretations of the same compositional syntax. For stationary kernels, spectral compilation maps kernel sums to measure addition and kernel products to convolution. For finite-rank kernels, feature compilation maps kernel sums to feature concatenation and kernel products to tensor products. In each case, the target semiring determines how larger compositions are evaluated once the primitive representations are specified. This viewpoint provides the algebraic basis for the extensibility of SPECTOR: incorporating a new primitive requires defining its representation in an appropriate target space, after which the corresponding semiring operations determine its behavior inside larger compositional programs.

Realization semantics is tree-level. The universal property above applies to interpretations whose target is a commutative semiring, which covers Eval_K, $\mathrm{Eval}_{\mathcal{S}}$, and $\mathrm{Eval}_{\mathcal{F}}$. It does not apply to the realization semantics Eval_R of Proposition 2.1: the law of the realized path depends on the tree, not only on its class in $\mathbb{N}[\mathcal{B}]$. For instance, $(f_1+f_2)f_3$ and $f_1f_3'+f_2f_3''$, where f_3' and f_3'' are independent copies of f_3, both have covariance $(k_1+k_2)k_3$ but differ in their fourth-order moments. Accordingly, Eval_R is defined on the term algebra $\mathfrak{T}(\mathcal{B})$, and it is only its covariance, $\mathrm{Cov} \circ \mathrm{Eval}_R = \mathrm{Eval}_K$, that descends to $\mathbb{N}[\mathcal{B}]$.

D Complete Theorem Statements and Proofs

D.1 Proposition 2.1

Proposition (Proposition 2.1 (Covariance consistency of realization trees))**.** *Let $\mathcal{T}$ be any finite sum-product expression whose leaf occurrences are assigned independent zero-mean realizations with the corresponding covariance kernels. If $\mathrm{Eval}_R(\mathcal{T})$ evaluates the same expression using $\oplus$ and $\odot$ in realization space, then $\mathrm{Eval}_R(\mathcal{T}) \in \mathcal{R}$ and $\mathrm{Cov}[\mathrm{Eval}_R(\mathcal{T})] = \mathrm{Eval}_K(\mathcal{T})$.*

Remark D.1. Proposition 2.1 relates the two evaluation maps on expression trees (the term algebra $\mathfrak{T}(\mathcal{B})$ of Appendix C); it is not a semiring homomorphism, and $(\mathcal{R}, \oplus, \odot)$ is not a semiring. Indeed $\mathcal{R}$ is not closed under $\odot$, since $f \odot f$ has mean $\mathrm{Var}\, f \neq 0$; it has no multiplicative identity, since the constant process $\mathbf{1}$ is not zero-mean; and Cov is multiplicative only on independent pairs, so it is not multiplicative on $\mathcal{R}$. The content of the proposition is instead the commuting identity $\mathrm{Cov} \circ \mathrm{Eval}_R = \mathrm{Eval}_K$ on $\mathfrak{T}(\mathcal{B})$, which holds precisely because distinct leaf occurrences receive independent realizations. Appendix C.3 discusses why this identity descends to $\mathbb{N}[\mathcal{B}]$ while Eval_R itself does not.

Proof. We proceed by structural induction on $\mathcal{T}$, simultaneously showing that every realized subtree lies in $\mathcal{R}$ and has covariance equal to its kernel-space evaluation. For a leaf k_ℓ, the claim holds by assumption: its assigned realization $f_\ell \in \mathcal{R}$ is zero-mean and satisfies

$$\mathrm{Cov}[f_\ell] = k_\ell.$$

Now suppose the claim holds for two child expressions $\mathcal{T}_a$ and $\mathcal{T}_b$. Let

$$f_a = \mathrm{Eval}_R(\mathcal{T}_a), \qquad f_b = \mathrm{Eval}_R(\mathcal{T}_b),$$

with corresponding kernels

$$k_a = \mathrm{Eval}_K(\mathcal{T}_a), \qquad k_b = \mathrm{Eval}_K(\mathcal{T}_b).$$

Because the two subtrees depend on disjoint sets of independent leaf occurrences, f_a and f_b are independent. By the induction hypothesis, $f_a, f_b \in \mathcal{R}$.

If the parent is a sum node, then $f = f_a \oplus f_b = f_a + f_b$. Hence $\mathbb{E}[f(t)] = 0$, and

$$\mathbb{E}[f(t)^2] \le 2\mathbb{E}[f_a(t)^2] + 2\mathbb{E}[f_b(t)^2] < \infty,$$

so $f \in \mathcal{R}$. Independence and zero means give

$$\mathrm{Cov}\big(f(t), f(t')\big) = k_a(t,t') + k_b(t,t'),$$

which is exactly the kernel-space evaluation of the corresponding sum node.

If the parent is a product node, then $f = f_a \odot f_b = f_a f_b$. Independence and zero means imply $\mathbb{E}[f(t)] = 0$, while

$$\mathbb{E}[f(t)^2] = \mathbb{E}[f_a(t)^2]\,\mathbb{E}[f_b(t)^2] < \infty,$$

so again $f \in \mathcal{R}$. Moreover, the products below are integrable by Cauchy–Schwarz, and independence yields:

$$\begin{aligned} \mathrm{Cov}\big(f(t), f(t')\big) &= \mathbb{E}\left[f_a(t) f_a(t') f_b(t) f_b(t')\right] && (5)\\ &= \mathbb{E}\left[f_a(t) f_a(t')\right] \mathbb{E}\left[f_b(t) f_b(t')\right] && (6)\\ &= k_a(t,t')\, k_b(t,t'). && (7) \end{aligned}$$

Thus the covariance equals the kernel-space evaluation of the corresponding product node. Therefore, by structural induction over the finite expression tree,

$$\mathrm{Eval}_R(\mathcal{T}) \in \mathcal{R}, \qquad \mathrm{Cov}[\mathrm{Eval}_R(\mathcal{T})] = \mathrm{Eval}_K(\mathcal{T}).$$

□

D.2 Proof of Proposition 2.2

Proposition (Proposition 2.2 (Spectral isomorphism for compositional kernel trees))**.** *Let $\mathcal{K}_{\mathrm{stat}}$ be the commutative semiring of real-valued continuous stationary positive-semidefinite kernels on $\mathbb{R}$ under addition and pointwise multiplication. Let $\mathcal{M}_+$ be the commutative semiring of finite nonnegative* symmetric *measures on $\mathbb{R}$ under measure addition and convolution. By Bochner's theorem, the spectral map*

$$\mathcal{S} : (\mathcal{K}_{\mathrm{stat}}, +, \cdot) \longrightarrow (\mathcal{M}_+, +, *), \qquad k(\tau) = \int_{\mathbb{R}} \cos(\omega\tau)\, d\mathcal{S}(k)(\omega),$$

is a semiring isomorphism: it is a bijection, it satisfies $\mathcal{S}(k_1 + k_2) = \mathcal{S}(k_1) + \mathcal{S}(k_2)$ and $\mathcal{S}(k_1 k_2) = \mathcal{S}(k_1) * \mathcal{S}(k_2)$, and it matches the identities, carrying the zero kernel to the zero measure and the constant kernel $k \equiv 1$ to δ_0. Consequently, for any finite sum-product expression tree $\mathcal{T}$ whose leaves belong to $\mathcal{K}_{\mathrm{stat}}$,

$$\mathcal{S}(\mathrm{Eval}_K(\mathcal{T})) = \mathrm{Eval}_{\mathcal{S}}(\mathcal{T}),$$

where $\mathrm{Eval}_{\mathcal{S}}$ evaluates $\mathcal{T}$ in $(\mathcal{M}_+, +, *)$ with each leaf replaced by its spectral measure.

Proof. By Bochner's theorem, every real-valued continuous stationary positive-semidefinite kernel $k \in \mathcal{K}_{\mathrm{stat}}$ admits a unique finite nonnegative symmetric spectral measure $\mu_k \in \mathcal{M}_+$ such that

$$k(\tau) = \int_{\mathbb{R}} e^{i\omega\tau}\, d\mu_k(\omega) = \int_{\mathbb{R}} \cos(\omega\tau)\, d\mu_k(\omega).$$

Conversely, for any $\mu \in \mathcal{M}_+$ the function $\tau \mapsto \int_{\mathbb{R}} \cos(\omega\tau)\, d\mu(\omega)$ is real-valued by symmetry of μ, continuous by dominated convergence since μ is finite and the integrand is bounded, and positive semidefinite; it therefore lies in $\mathcal{K}_{\mathrm{stat}}$ and has spectral measure μ. Uniqueness of μ_k gives injectivity, so $\mathcal{S}$ is a bijection. Restricting $\mathcal{M}_+$ to symmetric measures is necessary here: δ_{ω_0} with $\omega_0 \neq 0$ is the spectral measure of $e^{i\omega_0\tau}$, which is not a real-valued kernel and hence has no preimage in $\mathcal{K}_{\mathrm{stat}}$.

For addition, linearity of the integral gives

$$k_1(\tau) + k_2(\tau) = \int_{\mathbb{R}} e^{i\omega\tau}\, d\big(\mu_{k_1} + \mu_{k_2}\big)(\omega),$$

and therefore

$$\mathcal{S}(k_1 + k_2) = \mathcal{S}(k_1) + \mathcal{S}(k_2).$$

For pointwise multiplication, Fubini's theorem applies because $\mu_{k_1} \otimes \mu_{k_2}$ is a finite measure and the integrand has modulus one. Writing $\mu_{k_1} * \mu_{k_2}$ for the pushforward of $\mu_{k_1} \otimes \mu_{k_2}$ under $(\omega_1, \omega_2) \mapsto \omega_1 + \omega_2$,

$$k_1(\tau)k_2(\tau) = \int_{\mathbb{R}}\int_{\mathbb{R}} e^{i(\omega_1+\omega_2)\tau}\, d\mu_{k_1}(\omega_1)\, d\mu_{k_2}(\omega_2) \tag{8}$$

$$= \int_{\mathbb{R}} e^{i\omega\tau}\, d(\mu_{k_1} * \mu_{k_2})(\omega), \tag{9}$$

so

$$\mathcal{S}(k_1k_2) = \mathcal{S}(k_1) * \mathcal{S}(k_2).$$

Addition and convolution preserve symmetry. Moreover,

$$\mathcal{S}(0) = 0, \qquad \mathcal{S}(1) = \delta_0,$$

so $\mathcal{S}$ preserves the additive and multiplicative identities. Thus $\mathcal{S}$ is an isomorphism of commutative semirings.

It remains to establish the statement for expression trees. We proceed by structural induction on $\mathcal{T}$. If $\mathcal{T}$ consists of a single stationary kernel k, then

$$\mathcal{S}(\mathrm{Eval}_K(\mathcal{T})) = \mathcal{S}(k) = \mathrm{Eval}_{\mathcal{S}}(\mathcal{T}).$$

Now suppose the claim holds for two child expressions $\mathcal{T}_a$ and $\mathcal{T}_b$. If

$$\mathcal{T} = \mathrm{Sum}(\mathcal{T}_a, \mathcal{T}_b),$$

then

$$\mathcal{S}\left(\mathrm{Eval}_K(\mathcal{T})\right) = \mathcal{S}\left(\mathrm{Eval}_K(\mathcal{T}_a) + \mathrm{Eval}_K(\mathcal{T}_b)\right) \tag{10}$$

$$= \mathcal{S}\left(\mathrm{Eval}_K(\mathcal{T}_a)\right) + \mathcal{S}\left(\mathrm{Eval}_K(\mathcal{T}_b)\right) \tag{11}$$

$$= \mathrm{Eval}_{\mathcal{S}}(\mathcal{T}_a) + \mathrm{Eval}_{\mathcal{S}}(\mathcal{T}_b) \tag{12}$$

$$= \mathrm{Eval}_{\mathcal{S}}(\mathcal{T}). \tag{13}$$

If

$$\mathcal{T} = \mathrm{Prod}(\mathcal{T}_a, \mathcal{T}_b),$$

then

$$\mathcal{S}\left(\mathrm{Eval}_K(\mathcal{T})\right) = \mathcal{S}\left(\mathrm{Eval}_K(\mathcal{T}_a)\,\mathrm{Eval}_K(\mathcal{T}_b)\right) \tag{14}$$

$$= \mathcal{S}\left(\mathrm{Eval}_K(\mathcal{T}_a)\right) * \mathcal{S}\left(\mathrm{Eval}_K(\mathcal{T}_b)\right) \tag{15}$$

$$= \mathrm{Eval}_{\mathcal{S}}(\mathcal{T}_a) * \mathrm{Eval}_{\mathcal{S}}(\mathcal{T}_b) \tag{16}$$

$$= \mathrm{Eval}_{\mathcal{S}}(\mathcal{T}). \tag{17}$$

Therefore, by structural induction,

$$\mathcal{S}(\mathrm{Eval}_K(\mathcal{T})) = \mathrm{Eval}_{\mathcal{S}}(\mathcal{T})$$

for every finite sum-product expression tree $\mathcal{T}$. □

D.3 THEOREM 2.3

Theorem (Theorem 2.3 (Spectral tree compilation)). *Let $\mathcal{T}$ be any finite sum-product expression whose leaves are stationary kernels $k_1, \dots, k_L$, and let $k = \mathrm{Eval}_K(\mathcal{T})$. Let $\mu_1, \dots, \mu_L$ denote the corresponding spectral measures. Assume $k_\ell(0) > 0$ and define the normalized leaf spectral laws $p_\ell(d\omega) = d\mu_\ell(\omega)/k_\ell(0)$. Then the normalized spectral law of the composite kernel can be sampled exactly by a recursively constructed program $\mathcal{P}_\mathcal{T}$:*

1. ***Leaf***(k_ℓ)*: draw $\omega \sim p_\ell$.*
2. ***Sum***(a, b)*: draw $\omega \sim \mathcal{P}_a$ with probability $\pi_a = k_a(0)/(k_a(0) + k_b(0))$, and otherwise draw $\omega \sim \mathcal{P}_b$, where $k_a = \mathrm{Eval}_K(a)$ and $k_b = \mathrm{Eval}_K(b)$.*
3. ***Product***(a, b)*: independently draw $\omega_a \sim \mathcal{P}_a$ and $\omega_b \sim \mathcal{P}_b$, and return $\omega = \omega_a + \omega_b$.*

The output of $\mathcal{P}_\mathcal{T}$ has distribution

$$p_\mathcal{T}(d\omega) = d\mu_\mathcal{T}(\omega)/k(0), \qquad \mu_\mathcal{T} = \mathrm{Eval}_\mathcal{S}(\mathcal{T}).$$

Proof. We proceed by structural induction on the expression tree.

Base case (Leaf). For a leaf k_ℓ, Bochner's representation gives $\mu_\ell(\mathbb{R}) = k_\ell(0)$. Hence $p_\ell(d\omega) = d\mu_\ell(\omega)/k_\ell(0)$ is a probability measure, which is exactly the law sampled by $\mathcal{P}_{k_\ell}$.

Inductive step (Sum). Suppose $\mathcal{P}_a$ and $\mathcal{P}_b$ sample from $p_a = d\mu_a/k_a(0)$ and $p_b = d\mu_b/k_b(0)$, respectively. For the sum kernel $k_+ = k_a + k_b$, we have $\mu_+ = \mu_a + \mu_b$ and $k_+(0) = k_a(0) + k_b(0)$. The mixture rule therefore induces

$$\begin{aligned} p_+ &= \frac{k_a(0)}{k_a(0) + k_b(0)}\, p_a + \frac{k_b(0)}{k_a(0) + k_b(0)}\, p_b \\ &= \frac{d\mu_a + d\mu_b}{k_a(0) + k_b(0)} = \frac{d\mu_+}{k_+(0)}. \end{aligned} \tag{18}$$

Inductive step (Product). Suppose again that $\mathcal{P}_a$ and $\mathcal{P}_b$ sample from $p_a = d\mu_a/k_a(0)$ and $p_b = d\mu_b/k_b(0)$. For the product kernel $k_\times = k_a k_b$, we have $\mu_\times = \mu_a * \mu_b$ and $k_\times(0) = k_a(0)k_b(0)$. Since the sum of two independent random variables with laws p_a and p_b has law $p_a * p_b$, the product rule induces

$$p_\times = p_a * p_b = \frac{d(\mu_a * \mu_b)}{k_a(0)k_b(0)} = \frac{d\mu_\times}{k_\times(0)}. \tag{19}$$

Thus each recursive rule samples from the normalized spectral measure of the corresponding subtree, completing the induction. □

D.4 PROPOSITION D.2

Proposition D.2 ($O(MT)$ spectral sampler). *Given the compiled program $\mathcal{P}_\mathcal{T}$ for the composite kernel $k = \mathrm{Eval}_K(\mathcal{T})$, define*

$$f_M(t) = \sqrt{\frac{2\,k(0)}{M}} \sum_{m=1}^{M} \cos(\omega_m t + b_m), \tag{20}$$

where $\omega_1, \dots, \omega_M \overset{\text{iid}}{\sim} \mathcal{P}_\mathcal{T}$ and $b_1, \dots, b_M \overset{\text{iid}}{\sim} \mathrm{Unif}[0, 2\pi]$. Then: (i) For every finite M, $\mathbb{E}[f_M(t) f_M(t')] = k(t - t')$ for all t, t'; (ii) as $M \to \infty$, the finite-dimensional distributions of f_M converge to those of the unique zero-mean GP $\mathcal{GP}(0, k)$. In particular, for any fixed $t_1, \dots, t_d$, $\big(f_M(t_1), \dots, f_M(t_d)\big) \xrightarrow{d} \mathcal{N}(0, K_d)$ and $(K_d)_{ij} = k(t_i - t_j)$; (iii) for fixed d and nonsingular K_d, this finite-dimensional convergence admits a Berry-Esseen rate $O(M^{-1/2})$, with a constant depending on d, the evaluation points, and K_d; and (iv) the total cost is $O(nM + MT)$, where n is the number of leaves.

Proof. Let $k = \mathrm{Eval}_K(\mathcal{T})$ and let $p_\mathcal{T}(d\omega) = d\mu_\mathcal{T}(\omega)/k(0)$ be the output law of $\mathcal{P}_\mathcal{T}$ from Theorem 2.3.

(i) Exact covariance. For a single feature, with $\omega \sim p_{\mathcal{T}}$ and $b \sim \mathrm{Unif}[0, 2\pi]$, the product-to-sum identity gives

$$\mathbb{E}_b[\cos(\omega t + b)\cos(\omega t' + b)] = \tfrac{1}{2}\cos(\omega(t - t')). \tag{21}$$

Since $\omega \sim p_{\mathcal{T}}$ with $p_{\mathcal{T}}(d\omega) = d\mu_{\mathcal{T}}(\omega)/k(0)$,

$$\mathbb{E}_\omega[\cos(\omega(t - t'))] = \int_{\mathbb{R}} \cos(\omega(t - t'))\, p_{\mathcal{T}}(d\omega) \tag{22}$$

$$= \frac{1}{k(0)} \int_{\mathbb{R}} \cos(\omega(t - t'))\, d\mu_{\mathcal{T}}(\omega). \tag{23}$$

Because k is real-valued and stationary, its spectral measure is symmetric, and Bochner's representation reduces to

$$k(\tau) = \int_{\mathbb{R}} \cos(\omega\tau)\, d\mu_{\mathcal{T}}(\omega). \tag{24}$$

Setting $\tau = t - t'$ therefore gives

$$\mathbb{E}_\omega[\cos(\omega(t - t'))] = \frac{k(t - t')}{k(0)}. \tag{25}$$

Moreover, distinct features have zero cross-covariance because $\mathbb{E}_{b_m}[\cos(\omega_m t + b_m)] = 0$ and the pairs (ω_m, b_m) are independent. Therefore,

$$\mathbb{E}[f_M(t) f_M(t')] = \frac{2k(0)}{M} \sum_{m=1}^{M} \frac{1}{2} \frac{k(t - t')}{k(0)} = k(t - t'). \tag{26}$$

(ii) Finite-dimensional convergence. Fix arbitrary time points $t_1, \ldots, t_r$ and let

$$\mathbf{f}_M = \big(f_M(t_1), \ldots, f_M(t_r)\big)^\top. \tag{27}$$

To prove convergence of $\mathbf{f}_M$, consider an arbitrary vector $c = (c_1, \ldots, c_r)^\top \in \mathbb{R}^r$ and the scalar projection

$$c^\top \mathbf{f}_M = \sum_{j=1}^{r} c_j f_M(t_j) = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} Y_m, \tag{28}$$

where

$$Y_m = \sqrt{2k(0)} \sum_{j=1}^{r} c_j \cos(\omega_m t_j + b_m). \tag{29}$$

Because (ω_m, b_m) are i.i.d. across m, the variables $Y_1, \ldots, Y_M$ are i.i.d. Moreover, $\mathbb{E}[Y_m] = 0$ since $\mathbb{E}_{b_m}[\cos(\omega_m t_j + b_m)] = 0$, and Y_m is bounded for fixed c and r, hence has finite variance.

Using the covariance identity from part (i),

$$\mathrm{Var}(Y_m) = 2k(0) \sum_{j,\ell=1}^{r} c_j c_\ell \mathbb{E}[\cos(\omega_m t_j + b_m)\cos(\omega_m t_\ell + b_m)] \tag{30}$$

$$= \sum_{j,\ell=1}^{r} c_j c_\ell k(t_j - t_\ell). \tag{31}$$

Let $K_r \in \mathbb{R}^{r \times r}$ denote the covariance matrix with $(K_r)_{j\ell} = k(t_j - t_\ell)$. Then

$$\mathrm{Var}(Y_m) = c^\top K_r c. \tag{32}$$

Therefore, by the classical central limit theorem,

$$c^\top \mathbf{f}_M = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} Y_m \xrightarrow{d} \mathcal{N}\big(0, c^\top K_r c\big)\,. \tag{33}$$

Now let $\mathbf{Z} \sim \mathcal{N}(0, K_r)$. For every $c \in \mathbb{R}^r$,

$$c^\top \mathbf{Z} \sim \mathcal{N}\big(0, c^\top K_r c\big)\,. \tag{34}$$

Hence every scalar projection $c^\top \mathbf{f}_M$ converges in distribution to the corresponding scalar projection $c^\top \mathbf{Z}$. By the Cramér–Wold theorem, this implies

$$\mathbf{f}_M \xrightarrow{d} \mathbf{Z} \sim \mathcal{N}(0, K_r). \tag{35}$$

Since the choice of the finite set $t_1, \ldots, t_r$ was arbitrary, the finite-dimensional distributions of f_M converge to those of $\mathcal{GP}(0, k)$.

(iii) Finite-dimensional convergence rate. Fix $t_1, \ldots, t_r$ and assume that K_r is nonsingular. Define the i.i.d. random vectors

$$X_m = \sqrt{2\,k(0)}\,(\cos(\omega_m t_1 + b_m), \ldots, \cos(\omega_m t_r + b_m))^\top \tag{36}$$

so that

$$\mathbf{f}_M = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} X_m. \tag{37}$$

By part (i),

$$\mathbb{E}[X_m] = 0, \qquad \mathrm{Cov}(X_m) = K_r. \tag{38}$$

Whiten the summands by setting

$$W_m = K_r^{-1/2} X_m. \tag{39}$$

Then $\mathbb{E}[W_m] = 0$ and $\mathrm{Cov}(W_m) = I_r$. Moreover, since $|\cos(\cdot)| \le 1$,

$$\|X_m\|_2 \le \sqrt{2k(0)r}, \tag{40}$$

and hence

$$\mathbb{E}\|W_m\|_2^3 \le \left(\frac{2k(0)r}{\lambda_{\min}(K_r)}\right)^{3/2} < \infty. \tag{41}$$

A multivariate Berry-Esseen bound (Bentkus, 2003) therefore gives, for the class $\mathcal{C}_r$ of convex Borel sets in $\mathbb{R}^r$,

$$\sup_{A \in \mathcal{C}_r} \left| \Pr\left(\frac{1}{\sqrt{M}} \sum_{m=1}^{M} W_m \in A \right) - \Pr(Z \in A) \right| \le \frac{C\, r^{1/4}}{\sqrt{M}}\, \mathbb{E}\|W_1\|_2^3. \tag{42}$$

where $Z \sim \mathcal{N}(0, I_r)$ and C is a universal constant. Since multiplication by the invertible matrix $K_r^{1/2}$ preserves convexity of sets, the same $M^{-1/2}$ rate applies to $\mathbf{f}_M$ relative to $\mathcal{N}(0, K_r)$. Thus, for fixed r and fixed evaluation points,

$$\sup_{A \in \mathcal{C}_r} |\Pr(\mathbf{f}_M \in A) - \Pr(\mathbf{Z} \in A)| = O(M^{-1/2}), \qquad \mathbf{Z} \sim \mathcal{N}(0, K_r). \tag{43}$$

where the constant depends on r, the evaluation points through K_r, and $k(0)$. Thus, for any fixed finite set of evaluation points, the induced finite-dimensional distribution converges to the corresponding Gaussian-process marginal in convex-set distance at the Berry-Esseen rate $O(M^{-1/2})$, and therefore also converges weakly.

(iv) Complexity. Drawing one frequency from $\mathcal{P}_\mathcal{T}$ requires at most one traversal of an expression tree with n leaves: sum nodes recurse into one selected child, whereas product nodes recurse into both children and add the resulting frequencies. Thus, M frequency draws cost $O(nM)$ in the worst case. Evaluating the M spectral features at T time points costs $O(MT)$, giving total complexity $O(nM + MT)$. When n is bounded independently of T, as in our generator, this is $O(MT)$ in sequence length. □

D.5 Proof of Proposition 2.4

Proposition (Proposition 2.4 (Finite-rank feature homomorphism))**.** *Fix an index set $\mathcal{X}$ and let $\mathcal{K}_{\mathrm{FR}}$ be the commutative semiring of finite-rank positive-semidefinite kernels on $\mathcal{X}$ under addition and pointwise multiplication. Let*

$$\mathcal{F}_{\mathrm{FR}} = \left\{ \psi : \mathcal{X} \to \mathbb{R}^r \;:\; r \in \mathbb{N}_0 \right\} / \sim, \qquad \psi \sim \psi' \iff r = r' \text{ and } \psi' = Q\psi \text{ for some } Q \in O(r)$$

denote finite-dimensional feature maps modulo orthogonal equivalence, including the empty map $\varnothing : \mathcal{X} \to \mathbb{R}^0$. *Equip* $\mathcal{F}_{\mathrm{FR}}$ *with feature concatenation and the tensor-product feature map,*

$$(\psi_1 \oplus \psi_2)(t) = \begin{bmatrix} \psi_1(t) \\ \psi_2(t) \end{bmatrix}, \qquad (\psi_1 \otimes \psi_2)(t) = \psi_1(t) \otimes \psi_2(t),$$

and define

$$\kappa : \mathcal{F}_{\mathrm{FR}} \longrightarrow \mathcal{K}_{\mathrm{FR}}, \qquad \kappa\big([\psi]\big)(t, t') = \psi(t)^\top \psi(t').$$

Then $\oplus$, $\otimes$, *and* κ *are well defined on orthogonal-equivalence classes,* $(\mathcal{F}_{\mathrm{FR}}, \oplus, \otimes)$ *is a commutative semiring with additive identity* $[\varnothing]$ *and multiplicative identity* $[\mathbf{1}]$, $\mathbf{1}(t) \equiv 1$, *and*

$$\kappa : (\mathcal{F}_{\mathrm{FR}}, \oplus, \otimes) \longrightarrow (\mathcal{K}_{\mathrm{FR}}, +, \cdot)$$

is a surjective semiring homomorphism. Consequently, for any finite sum-product expression tree $\mathcal{T}$ *whose leaves are assigned finite-rank feature representations,*

$$\kappa(\mathrm{Eval}_{\mathcal{F}}(\mathcal{T})) = \mathrm{Eval}_K(\mathcal{T}).$$

Corollary D.3 (Compiled feature dimension)**.** *The dimension map* $\dim : [\psi] \mapsto r$ *is a semiring homomorphism* $(\mathcal{F}_{\mathrm{FR}}, \oplus, \otimes) \to (\mathbb{N}_0, +, \times)$. *Hence, if leaf* ℓ *is assigned a representation of dimension* r_ℓ, *the compiled representation of* $\mathcal{T}$ *has dimension* $\mathrm{Eval}_{\mathbb{N}_0}(\mathcal{T})$, *the tree evaluated in* $(\mathbb{N}_0, +, \times)$ *with each leaf replaced by* r_ℓ, *and*

$$\operatorname{rank} \mathrm{Eval}_K(\mathcal{T}) \;\le\; \dim \mathrm{Eval}_{\mathcal{F}}(\mathcal{T}) = \mathrm{Eval}_{\mathbb{N}_0}(\mathcal{T}).$$

The inequality can be strict: for $\psi(t) = (1, t)^\top$ *and* $k = \kappa([\psi])$, *the product* $\psi \otimes \psi$ *has dimension* 4 *while* $\operatorname{rank}(k^2) = 3$.

Proof. We first show that the operations and the map κ are well defined on orthogonal-equivalence classes. Suppose $\psi_j' = Q_j \psi_j$ with $Q_j \in O(r_j)$ for $j \in \{1, 2\}$. Then

$$\psi_1' \oplus \psi_2' = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} (\psi_1 \oplus \psi_2),$$

and the block-diagonal matrix is orthogonal. Hence concatenation is well defined on equivalence classes. Likewise,

$$\psi_1' \otimes \psi_2' = (Q_1 \otimes Q_2)(\psi_1 \otimes \psi_2),$$

and $Q_1 \otimes Q_2$ is orthogonal, so tensor-product composition is also well defined. Moreover, if $\psi' = Q\psi$ with Q orthogonal, then

$$\psi'(t)^\top \psi'(t') = \psi(t)^\top Q^\top Q \psi(t') = \psi(t)^\top \psi(t'),$$

so κ is well defined.

We next verify the semiring structure. Feature concatenation is associative and commutative up to orthogonal coordinate permutations, and has the empty feature map $[\varnothing]$ as its identity. Tensor-product composition is likewise associative and commutative up to orthogonal coordinate permutations, with $[\mathbf{1}]$ as its identity. It distributes over concatenation because

$$\psi_1 \otimes (\psi_2 \oplus \psi_3)$$

differs from

$$(\psi_1 \otimes \psi_2) \oplus (\psi_1 \otimes \psi_3)$$

only by a permutation of coordinates. Finally, $\varnothing \otimes \psi = \varnothing$, so the additive identity is absorbing under multiplication. Therefore $(\mathcal{F}_{\mathrm{FR}}, \oplus, \otimes)$ is a commutative semiring.

Now let $k_j = \kappa([\psi_j])$. For addition,

$$\kappa([\psi_1] \oplus [\psi_2])(t, t') = (\psi_1 \oplus \psi_2)(t)^\top (\psi_1 \oplus \psi_2)(t') \tag{44}$$

$$= \psi_1(t)^\top \psi_1(t') + \psi_2(t)^\top \psi_2(t') \tag{45}$$

$$= k_1(t, t') + k_2(t, t'). \tag{46}$$

Hence

$$\kappa([\psi_1] \oplus [\psi_2]) = \kappa([\psi_1]) + \kappa([\psi_2]).$$

For multiplication, the Kronecker-product identity

$$(a \otimes b)^\top (c \otimes d) = (a^\top c)(b^\top d)$$

gives

$$\kappa([\psi_1] \otimes [\psi_2])(t, t') = (\psi_1(t) \otimes \psi_2(t))^\top (\psi_1(t') \otimes \psi_2(t')) \tag{47}$$

$$= \big(\psi_1(t)^\top \psi_1(t')\big)\big(\psi_2(t)^\top \psi_2(t')\big) \tag{48}$$

$$= k_1(t, t')\, k_2(t, t'). \tag{49}$$

Therefore

$$\kappa([\psi_1] \otimes [\psi_2]) = \kappa([\psi_1])\, \kappa([\psi_2]).$$

The identities are preserved:

$$\kappa([\varnothing]) = 0, \qquad \kappa([\mathbf{1}]) = 1.$$

Thus κ is a semiring homomorphism.

By definition, every finite-rank positive-semidefinite kernel admits a finite-dimensional feature representation $k(t, t') = \psi(t)^\top \psi(t')$. Hence κ is surjective.

Finally, consider any finite sum-product expression tree $\mathcal{T}$ whose leaves are assigned finite-dimensional feature maps. Since κ preserves both semiring operations, structural induction on $\mathcal{T}$ gives

$$\kappa(\mathrm{Eval}_{\mathcal{F}}(\mathcal{T})) = \mathrm{Eval}_K(\mathcal{T}).$$

□

D.6 Proof of Proposition 2.5

Proposition (Proposition 2.5 (Gaussian modulation))**.** *Let $k_{\mathrm{FR}}(t, t') = \sum_{j=1}^r \psi_j(t)\psi_j(t')$ be a finite-rank kernel and let k_g be any kernel admitting a zero-mean Gaussian-process realization. If $g_1, \ldots, g_r \overset{\mathrm{ind}}{\sim} \mathcal{GP}(0, k_g)$ and*

$$f(t) = \sum_{j=1}^r \psi_j(t) g_j(t), \tag{50}$$

then

$$f \sim \mathcal{GP}(0, k_{\mathrm{FR}} k_g), \qquad \mathrm{Cov}(f(t), f(t')) = k_{\mathrm{FR}}(t, t') k_g(t, t'). \tag{51}$$

Proof. For any finite set of time points, f is a linear transformation of the jointly Gaussian variables $\{g_j(t_i)\}$ and is therefore Gaussian. Moreover,

$$\mathrm{Cov}(f(t), f(t')) = \sum_{j,\ell=1}^r \psi_j(t)\phi_\ell(t')\, \mathrm{Cov}(g_j(t), g_\ell(t')) \tag{52}$$

$$= \sum_{j=1}^r \psi_j(t)\psi_j(t') k_g(t, t') \tag{53}$$

$$= k_{\mathrm{FR}}(t, t') k_g(t, t'), \tag{54}$$

where the second equality follows from independence of the g_j. □

E Examples of SPECTOR Realization Operators

This section gives concrete examples of the realization operators used by SPECTOR and verifies their covariance properties. We distinguish *law-preserving* realizations, which reproduce both the target covariance and Gaussian-process law, from *covariance-preserving* realizations, which reproduce the target kernel exactly but may induce a non-Gaussian random-function prior.

E.1 Finite-Rank Kernels

For any finite-rank kernel $k(t,t') = \psi(t)^\top \psi(t')$ with $\psi(t) \in \mathbb{R}^r$, drawing $z \sim \mathcal{N}(0, I_r)$ and setting $f(t) = \psi(t)^\top z$ yields

$$\mathrm{Cov}(f(t), f(t')) = \psi(t)^\top \mathbb{E}[zz^\top]\psi(t') = k(t,t').$$

Because f is a linear transformation of Gaussian coefficients, this realization is exact and law-preserving.

Linear kernel. For $k_{\mathrm{lin}}(t,t') = \sigma_0^2 + \sigma_1^2 tt'$, take $\psi(t) = [\sigma_0, \sigma_1 t]^\top$ and $z = (z_0, z_1)^\top \sim \mathcal{N}(0, I_2)$. Then $f(t) = \sigma_0 z_0 + \sigma_1 z_1 t$, i.e., a random affine trend, and $\mathrm{Cov}(f(t), f(t')) = \sigma_0^2 + \sigma_1^2 tt'$.

Polynomial kernels. If a polynomial kernel admits the finite expansion $k_{\mathrm{poly}}(t,t') = \sum_{j=0}^{r-1} \psi_j(t)\psi_j(t')$, define $\psi(t) = [\phi_0(t), \ldots, \phi_{r-1}(t)]^\top$. The realization $f(t) = \sum_{j=0}^{r-1} z_j \phi_j(t)$ with $z_j \overset{\mathrm{iid}}{\sim} \mathcal{N}(0,1)$ is Gaussian and satisfies $\mathrm{Cov}(f(t), f(t')) = k_{\mathrm{poly}}(t,t')$ exactly.

Jacobi and Hermite kernels. The same construction applies when the feature map is formed from a finite collection of Jacobi or Hermite basis functions. If $k(t,t') = \sum_{j=0}^{r-1} \phi_j(t)\phi_j(t')$, with ϕ_j drawn from the corresponding polynomial family, then $f(t) = \sum_{j=0}^{r-1} z_j \phi_j(t)$ is an exact law-preserving realization. Thus these kernels can be sampled in $O(rT)$ time without materializing a $T \times T$ covariance matrix.

E.2 Recursive Infinite-Rank Realizations

Some nonstationary kernels are not finite rank but admit exact recursive realizations with fixed-dimensional Gaussian state.

Gaussian independent increments. Consider

$$f(t_i) = f(t_{i-1}) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, q_i),$$

with independent increments and $f(t_0) = 0$. Then $f(t_i) = \sum_{\ell=1}^{i} \epsilon_\ell$, and therefore

$$\mathrm{Cov}(f(t_i), f(t_j)) = \sum_{\ell=1}^{\min(i,j)} q_\ell = V(\min(t_i, t_j))\,,$$

where $V(t_i) = \sum_{\ell=1}^{i} q_\ell$. The resulting process is Gaussian, so this is an exact law-preserving realization.

Wiener/min kernel. Choosing $q_i = \sigma^2(t_i - t_{i-1})$ gives $V(t_i) = \sigma^2 t_i$ and hence

$$k_{\mathrm{W}}(t,t') = \sigma^2 \min(t,t').$$

Thus the Wiener, or min, kernel is sampled exactly by cumulative summation of independent Gaussian increments in $O(T)$ time.

Integrated Wiener processes. Higher-order integrated Wiener processes require a finite-dimensional latent state rather than a scalar recursion. In general, let

$$x_{i+1} = A_i x_i + \eta_i, \qquad \eta_i \sim \mathcal{N}(0, Q_i), \qquad f(t_i) = C_i x_i,$$

where A_i is the state-transition matrix, Q_i the innovation covariance, and C_i the projection from latent state to the observed scalar process. For integrated Wiener processes, the state contains the current process value together with the required derivative or integral components. Since the recursion is linear Gaussian, the resulting process is Gaussian and its induced covariance is reproduced exactly by sequential simulation.

Finite-dimensional Gaussian state-space models. Consider

$$x_{i+1} = A_i x_i + \eta_i, \qquad \eta_i \sim \mathcal{N}(0, Q_i), \qquad f(t_i) = C_i x_i,$$

with Gaussian initial state and independent innovations. This defines a Gaussian process whose covariance is induced by the state recursion and can be sampled exactly by sequential simulation, without constructing the corresponding $T \times T$ covariance matrix. For fixed state dimension d, sampling costs $O(Td^2)$ in general and $O(T)$ when d is fixed and the transition structure is constant-size.

E.3 Composition in Realization Space

SPECTOR combines independently realized child processes directly in sample-path space. The resulting guarantees depend on the composition rule.

Independent sums. Let f_a and f_b be independent zero-mean processes with covariance kernels k_a and k_b, and define $f_+(t) = f_a(t) + f_b(t)$. Then

$$\mathrm{Cov}(f_+(t), f_+(t')) = k_a(t,t') + k_b(t,t'),$$

since the cross terms vanish by independence and zero means. If both child processes are Gaussian, their sum is Gaussian; hence independent addition is law-preserving in the Gaussian case.

Finite-rank $\times$ Gaussian: Gaussian modulation. Let

$$k_{\mathrm{FR}}(t,t') = \sum_{j=1}^{r} \psi_j(t)\psi_j(t')$$

and let $g_j \overset{\mathrm{ind}}{\sim} \mathcal{GP}(0, k_g)$. Define

$$f(t) = \sum_{j=1}^{r} \psi_j(t) g_j(t).$$

Then

$$\mathrm{Cov}(f(t), f(t')) = \sum_{j=1}^{r} \psi_j(t)\psi_j(t') k_g(t,t') = k_{\mathrm{FR}}(t,t') k_g(t,t').$$

Because f is a linear combination of jointly GPs, it is itself Gaussian. Gaussian modulation therefore preserves both the product kernel and the full Gaussian-process law.

General independent products. Let f_a and f_b be independent zero-mean processes with covariance kernels k_a and k_b, and define $f_\times(t) = f_a(t) f_b(t)$. Then

$$\begin{aligned}\mathrm{Cov}(f_\times(t), f_\times(t')) &= \mathbb{E}[f_a(t) f_a(t') f_b(t) f_b(t')] \\ &= \mathbb{E}[f_a(t) f_a(t')]\mathbb{E}[f_b(t) f_b(t')] \\ &= k_a(t,t') k_b(t,t').\end{aligned}$$

Thus independent path multiplication preserves the target product kernel exactly. Unlike Gaussian modulation, however, it does not generally preserve Gaussian law.

Wiener $\times$ stationary. Let $B \sim \mathcal{GP}(0, k_{\mathrm{W}})$ with $k_{\mathrm{W}}(t,t') = \sigma^2 \min(t,t')$, and let $g \sim \mathcal{GP}(0, k_{\mathrm{stat}})$ independently. Setting $f(t) = B(t)g(t)$ gives

$$\mathrm{Cov}(f(t), f(t')) = \sigma^2 \min(t,t')\, k_{\mathrm{stat}}(t - t')$$

exactly. The marginals are generally product-normal rather than Gaussian, so this realization is covariance-preserving but not law-preserving.

Summary. These examples illustrate the hierarchy used by SPECTOR. Whenever possible, the generator uses a law-preserving realization—spectral synthesis in the large-M limit, exact finite-rank feature sampling, exact Gaussian state-space simulation, or Gaussian modulation. When a Gaussian-preserving product realization is unavailable, independent path multiplication provides an exact covariance-preserving alternative. In all cases, the realization is performed directly in frequency, feature, state, or sample-path space rather than through dense covariance factorization.

F SPECTOR IMPLEMENTATION DETAILS

This section provides the full implementation details of the SPECTOR configuration used in our empirical evaluation.

F.1 KERNEL BANK

The kernel bank comprises 18 base kernel types organized into four families by their sampling mechanism.

Stationary kernels (RBF, Matérn, RationalQuadratic, Constant) generate the slowly varying processes that form the backbone of most time series: their spectral mass concentrates at low frequencies (the RBF spectrum is Gaussian and decays super-exponentially; Matérn spectra decay polynomially at a rate set by ν, giving explicit control over sample-path differentiability).

Periodic kernels (ExpSineSquared, PeriodicWaveform) produce seasonal and cyclical patterns. The addition of non-sinusoidal waveforms (square, triangle, sawtooth, comb) captures the sharp transitions seen in industrial demand, sensor duty cycles, and power systems.

Trend and growth kernels (DotProduct, Polynomial, Jacobi, Hermite, Wiener, IntegratedWiener) model nonstationary level shifts. Polynomial and orthogonal-basis kernels generate structured trends; Wiener and IntegratedWiener produce stochastic drift.

Regime and transient kernels (ChangePoint, Windowed, ArcCosine) produce structural breaks and localized events. When multiplied with stationary kernels, ChangePoint yields series with distinct regimes separated by smooth sigmoidal transitions, while Windowed produces transient oscillations confined to a temporal window—patterns absent from both the original KernelSynth bank and ARIMA-based generators like SarSim0.

Noise kernels. White noise and Student-t noise share the same flat-spectrum covariance; they differ in law, not kernel. A covariance function is blind to tail behavior, so heavy-tailed marginals cannot be obtained from any Gaussian covariance-based sampler; SPECTOR attaches the Student-t law through a direct time-domain sampler while preserving the white second-order structure.

Table 2 lists all kernels with their hyperparameter grids and sampling weights.

Spectral densities (stationary kernels). Stationary kernels are sampled via spectral compilation (Theorem 2.3). Their normalized spectral densities $p(\omega) = S(\omega)/k(0)$ are:

$$\text{RBF:} \quad p(\omega) = \mathcal{N}(\omega; 0, (2\pi\ell)^{-2}), \tag{55}$$

$$\text{Matérn:} \quad p(\omega) \propto \left(\frac{2\nu}{\ell^2} + 4\pi^2\omega^2\right)^{-(\nu+1/2)}, \tag{56}$$

$$\text{ESS:} \quad p(\omega) = \sum_{n=-\infty}^{\infty} I_n(\eta)\, e^{-\eta}\, \delta(\omega - n/P), \quad \eta = \ell^{-2}, \tag{57}$$

$$\text{RQ:} \quad p(\omega) \propto |\omega|^{\alpha-1/2} K_{\alpha-1/2}(2\pi\ell\sqrt{2\alpha}\,|\omega|), \tag{58}$$

$$\text{White:} \quad p(\omega) = \mathrm{Unif}[-\pi, \pi] \quad \text{(flat on the Nyquist band)}, \tag{59}$$

$$\text{Waveform:} \quad p(\omega) = \sum_{n=1}^{N_h} c_n\, \delta(\omega - n/P), \tag{60}$$

where ℓ is the length scale, P the period, I_n are modified Bessel functions, and c_n are the Fourier coefficients of the waveform shape.

The white-noise kernel $k(\tau) = \sigma^2\delta_{\tau,0}$, with $\delta_{\tau,0}$ the Kronecker delta on integer lags, is discontinuous at the origin and therefore not an element of $\mathcal{K}_{\mathrm{stat}}$ as a kernel on $\mathbb{R}$; a constant density on $\mathbb{R}$ is likewise not a finite measure. It is instead handled by the discrete-time (Herglotz) form of Bochner's theorem on the uniform sampling grid, where, with unit spacing, its spectral measure is $\sigma^2\,\mathrm{Unif}[-\pi, \pi]$, since

$$\frac{\sigma^2}{2\pi}\int_{-\pi}^{\pi} \cos(\omega\tau)\, d\omega = \sigma^2\,\frac{\sin(\pi\tau)}{\pi\tau} = \sigma^2\delta_{\tau,0} \qquad \text{for every integer } \tau.$$

Convolution at a product node can push sampled frequencies outside $[-\pi, \pi]$; this is immaterial on the grid, because $e^{i\omega\tau}$ is 2π-periodic in ω for integer τ, so out-of-band frequencies alias onto band-equivalent ones without changing the realized covariance.

Feature maps (finite-rank kernels). Nonstationary kernels with $k(t, t') = \psi(t)^\top \psi(t')$ are sampled via Gaussian modulation (Proposition 2.5). Their feature maps are:

$$\text{DotProduct:} \quad \psi(t) = [\sigma_0,\, t]^\top, \quad r = 2, \tag{61}$$

$$\text{Polynomial:} \quad \psi(t) = [1,\, t,\, t^2, \ldots, t^d]^\top, \quad r = d + 1, \tag{62}$$

$$\text{Windowed:} \quad \psi(t) = w(t; \mu, \sigma_w), \quad r = 1, \tag{63}$$

$$\text{ChangePoint:} \quad \psi(t) = [\sigma(a(t-c)),\, 1 - \sigma(a(t-c))]^\top, \quad r = 2, \tag{64}$$

where w is the window function (Hann, Tukey, or Gaussian), σ is the sigmoid, c is the changepoint location, and $a = 1/\text{width}$ controls sharpness.

Direct samplers. Wiener and IntegratedWiener kernels are sampled via cumulative summation in $O(T)$: $W(t) = \sigma \sum_{i=1}^{t} \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, 1/T)$, and integrated Wiener applies k successive cumulative sums. ArcCosine kernels use a random feature approximation with ReLU (order 1) or Heaviside (order 0) activation applied to random projections.

F.2 Kernel Composition

Expression trees are sampled as follows:

1. Draw the number of leaves $K \sim \text{Uniform}\{1, 2, 3, 4, 5\}$ (i.e., $n_K = 5$).
2. Sample K leaf kernels independently from the weighted kernel bank (Table 2).
3. Fold left-associatively: starting from the first leaf, at each step combine with the next leaf using addition ($+$) with probability $p_{\text{add}} = 0.5$ or multiplication ($\times$) with probability 0.5.

The resulting expression tree is then partitioned:

- **Stationary subtrees** (all leaves are stationary) are compiled into a single spectral sampling program via Theorem 2.3.
- **Mixed products** (stationary $\times$ finite-rank) are handled via Gaussian modulation (Proposition 2.5).
- **Remaining leaves** (Wiener, ArcCosine, Student-t) use their dedicated $O(T)$ samplers.

F.3 Sampling

Path synthesis uses $M = 64$ random Fourier features. For a compiled stationary expression with spectral program $\mathcal{P}$, the sample path is:

$$f(t) = \sqrt{\frac{2\,k(0)}{M}} \sum_{m=1}^{M} \cos(2\pi\omega_m t + b_m), \quad \omega_m \sim \mathcal{P},\ b_m \sim \text{Unif}[0, 2\pi]. \tag{65}$$

Figure 5 illustrates sampling procedure of an example tree with both stationary and non-stationary finite-rank kernels, showing how spectral tree compilation converts kernel expression trees into frequency draws and how Gaussian modulation handles nonstationary factors.

Generation is performed on-the-fly during training with no pre-materialized dataset, providing an effectively infinite and non-repeating pretraining corpus.

F.4 Augmentation

Generated series undergo a two-stage augmentation pipeline:

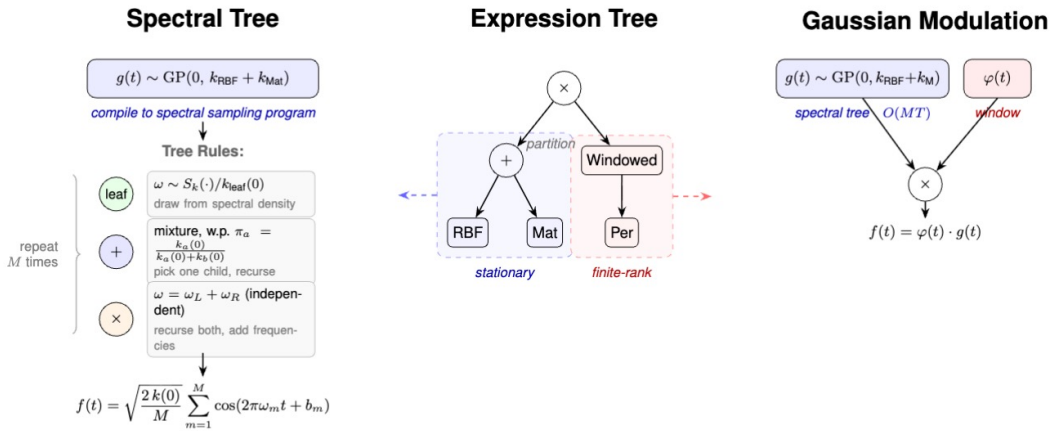


Figure 5: Sampling diagram illustrating spectral tree compilation and Gaussian modulation. Stationary kernel expression trees are compiled into spectral sampling programs that draw frequencies directly, while nonstationary finite-rank factors are incorporated via deterministic envelope modulation of independent GP draws.

Stage 1: Offline augmentation (SCM). Applied with probability 1.0. A Structural Causal Model (SCM) augmentation correlates independently generated series via random DAG propagation:

- A random DAG with $N_{\text{nodes}} = 8$ nodes and maximum in-degree 6 is sampled.
- Root nodes are assigned series from the generated batch.
- Non-root nodes compute $z = \sigma(\sum_j w_j \cdot \text{parent}_j + b)$, where σ is drawn uniformly from {linear, ReLU, sigmoid, sin, mod, leakyReLU} and $w_j \sim \mathcal{N}(0, 1)$.
- $N_{\text{channels}} = 4$ output nodes are selected per group, producing correlated series from independent GP draws.

Stage 2: Online augmentation (value transforms). Applied with probability $p = 0.5$. One of the following transforms is selected uniformly at random:

- **Quantization**: Snaps continuous values to $L \sim U(3, 20)$ non-equidistant levels uniformly distributed within the series range, simulating discrete/integer demand data.
- **SpikeInjection**: Adds periodic spike trains following TiRex Algorithm 3. Parameters: period $\pi \sim U(10, 512)$, kernel width $\sim U(1, 20)$, kernel shape $\in$ {tophat, RBF, triangle}, periodic pattern $\in$ {simple, 3-periodic, 4-periodic}, amplitude $\sim \text{LogNormal}(0, 1) \times 3\sigma_y$.
- **AmplitudeModulation**: Multiplies by a piecewise-linear envelope with $k \sim U(0, 5)$ random changepoints and amplitudes $a_i \sim \mathcal{N}(1, 1)$, producing non-stationary scale changes and heteroscedastic variance.

Table 2: Complete kernel bank used in SPECTOR. Weights determine the relative probability of selecting each kernel type as a leaf in the expression tree. Hyperparameters are drawn uniformly from the listed discrete grids. Blue entries are new additions beyond the legacy KernelSynth bank (Ansari et al., 2024).

Kernel	Class	Weight	Hyperparameters	Realization Space
Smooth, stationary: slowly varying, multi-scale, and controlled-roughness processes				
RBF	Stationary	3.0	length_scale $\in$ {20.48, 51.2, 102.4, 204.8, 358.4, 1024, 10240}/1024	Spectral
Matérn	Stationary	1.0	length_scale $\in$ {20.48, 51.2, 102.4, 204.8}/1024; $\nu \in$ {0.5, 1.5, 2.5}	Spectral
RationalQuadratic	Stationary	3.0	length_scale $\in$ {51.2, 102.4, 204.8, 1024}/1024; $\alpha \in$ {0.1, 1, 3, 10, 30}	Spectral
Constant	Stationary	1.0	—	Spectral
Periodic, stationary: smooth seasonal cycles and sharp-overtone periodicity				
ExpSineSquared	Stationary	21.0	period $\in$ {24, 48, 96, 168, 336, 672, 7, 14, 30, 60, 365, 730, 4, 26, 52, 6, 12, 40, 10}	Spectral tree
PeriodicWaveform	Stationary	1.0	shape $\in$ {square, triangle, sawtooth, comb}; period $\in$ {10.24, 20.48, 51.2, 102.4, 256}/1024	Spectral
Rough / noisy: uncorrelated Gaussian and heavy-tailed fluctuations				
White	Stationary	2.0	noise_level $\in$ {0.1, 1.0}	Spectral
Student t-dist)	Stationary	1.0	noise_level $\in$ {0.1, 1.0}; df $\in$ {3, 5, 8}	Time-domain
Trend / growth: linear, polynomial, stochastic, and nonlinear nonstationarity				
DotProduct	Non-Stationary Finite-Rank	3.0	$\sigma_0 \in$ {0, 1, 10}	Feature
Polynomial	Non-Stationary Finite-Rank	1.0	degree $\in$ {1, 2, 3, 4, 5}	Feature
Jacobi	Non-Stationary Finite-Rank	1.0	degree $\in$ {2, 3, 4, 5}; $a \in$ {0.5, 1, 2, 4}; $b \in$ {0.5, 1, 2, 4}	Feature
Hermite	Non-Stationary Finite-Rank	1.0	degree $\in$ {2, 3, 4, 5}; scale $\in$ {0.5, 1, 1.5, 2}	Feature
Wiener	Structured Infinite-Rank	5.0	$\sigma = 1.0$	Time-domain
IntegratedWiener	Structured Infinite-Rank	5.0	order $k \in \{1, 2\}$; $\sigma = 1.0$	Time-domain
ArcCosine	Structured Infinite-Rank	5.0	order $\in$ {0, 1}; $n_{\text{features}} \in$ {64, 128, 256}	Time-domain
Regime / transient: structural breaks, localized events, and modulated stochastic dynamics				
Windowed	Non-Stationary Finite-Rank	1.0	type $\in$ {hann, tukey, gauss}; $\mu \sim U(0.2, 0.8)$; param $\sim U(0.2, 0.7)$	Feature
ChangePoint	Non-Stationary Finite-Rank	1.0	location $\sim U(0.2, 0.8)$; width $\sim U(0.02, 0.1)$	Feature
Wiener$\times$Stationary	Structured Infinite-Rank	5.0	$\sigma = 1.0$	Time-domain

G PRETRAINING DETAILS

G.1 MODEL PRETRAINING AT $L = 1024$ FOR ALL SYNTHETIC GENERATORS

In this section, we describe in details the training configs used for each model for results in Table 1. All generators generate sequence of length $L = 1024$ and all share the same training protocol within each model.

Table 3: N-BEATS pretraining hyperparameters ($L = 1024$).

Hyperparameter	**Value**
Architecture	
Backbone	N-BEATS, generic blocks
Blocks / FC layers per block	10 / 3
Layer width	1024
Weight sharing across blocks	no
Input / output size	896 / 128 $\times$ 9 quantiles
Data	
Series length / horizon	1024 / 128
Training context	$\mathrm{Unif}[8, 896]$
Batch size	256 per GPU, 2 GPUs
Optimization	
Optimizer	Adam, lr 10^{-4}, no weight decay
LR schedule	10k warmup, cosine to $0.1\times$ peak
Steps	250k
Gradient clipping	0.1
Precision	bf16-mixed
Loss	pinball, quantiles $\{0.1, \ldots, 0.9\}$

Table 4: MQCNN pretraining hyperparameters ($L = 1024$).

Hyperparameter	**Value**
Architecture	
Backbone	MQ-CNN, univariate
Encoder	dilated causal Conv1d, kernel 2, dilations $1, 2, \ldots, 512$ (10 layers), residual connections
Encoder hidden size	128
Decoder hidden sizes	horizon-agnostic context 100, horizon-specific context 20, local decoder 20
Dropout	0.1
Input / output size	896 / 128 $\times$ 9 quantiles
Data	
Series length / horizon	1024 / 128
Training context	$\mathrm{Unif}[8, 896]$
Batch size	256 per GPU, 2 GPUs
Optimization	
Optimizer	AdamW, lr 10^{-3}, no weight decay
LR schedule	10k warmup, cosine to $0.1\times$ peak
Steps	250k
Gradient clipping	0.1
Precision	bf16-mixed
Loss	pinball, quantiles $\{0.1, \ldots, 0.9\}$

Table 5: Chronos-2 pretraining hyperparameters ($L = 1024$).

Hyperparameter	**Value**
Architecture	
Backbone	Chronos-2 base (T5-style encoder)
d_{model} / d_{ff} / d_{kv}	768 / 3072 / 64
Layers / heads	12 / 12
Feed-forward / positions	ReLU / RoPE ($\theta = 10^4$)
Dropout / init. factor	0.1 / 0.05
Patch size (input = output, stride)	16
Max. output patches	8 ($= H/16$)
Quantiles	21 levels: $0.01, 0.05, 0.1, 0.15, \ldots, 0.95, 0.99$
Data	
Context / horizon	1024 / 128
Min. context / value-drop prob.	64 / 0.2
Training mode	univariate
Batch size	32 per GPU, 8 GPUs (256)
Optimization	
Optimizer	AdamW (fused), lr 10^{-4}
LR schedule	cosine, 5% warmup
Steps	2M

Table 6: PatchTST-FM pretraining hyperparameters ($L = 1024$).

Hyperparameter	**Value**
Architecture	
Backbone	PatchTST-FM base (pre-norm encoder)
d_{model} / layers / heads	1024 / 20 / 16
Patch size	16
Quantile head	99 levels $\{0.01, \ldots, 0.99\}$
Context length	1024
Masking	ratio 0.4, contiguous blocks of 8 patches
Prediction tail mask	up to 128 steps ($= H$)
Data	
Series length / horizon	1024 / 128
Training context	$\text{Unif}[64, 896]$
Batch size	32 per GPU, 8 GPUs, $4\times$ accumulation (1024)
Optimization	
Optimizer	AdamW, lr $3 \cdot 10^{-4}$, $\beta = (0.9, 0.95)$, wd 0.1
LR schedule	5k warmup, cosine to $0.033\times$ peak
Steps	100k
Gradient clipping	1.0
Precision	fp16-mixed

Table 7: S4D pretraining hyperparameters ($L = 1024$).

Hyperparameter	**Value**
Architecture	
Backbone	S4D
Width / state size	512 / 128
Depth / expansion	6 / 4
Initialization	S4D-Lin
Dropout	0
Context / output	1024 / 128×9 quantiles
Data	
Series length / horizon	1024 / 128
Training context	$\mathrm{Unif}[8, 896]$
Batch size	32 per GPU, 8 GPUs (256)
Optimization	
Optimizer	AdamW, lr 10^{-3}, wd 0.03
LR schedule	5k warmup, cosine to $0.01\times$ peak
Steps	250k
Gradient clipping	1.0
Precision	bf16-mixed
Loss	pinball, quantiles $\{0.1, \ldots, 0.9\}$

Table 8: TiRex pretraining hyperparameters ($L = 1024$).

Hyperparameter	**Value**
Architecture	
Backbone	TiRex (xLSTM, sLSTM blocks), 35M parameters
Blocks / embedding dim / heads	12 / 512 / 4
Feed-forward dim	2048
Patch size (input = output)	32
Quantiles	9 levels $\{0.1, \ldots, 0.9\}$
Data	
Series length / training context	1024 / windows of 192–1024
Prediction tail mask	128 steps (fixed), ≥ 64 visible
Context masking	up to 25% of patches in up to 5 blocks
Batch size	256 (global)
Optimization	
Learning rate	10^{-3}, cosine to 10^{-4}, 5% warmup
Weight decay	0.01
Steps	500k
Gradient clipping	1.0
Precision	bf16

G.2 Model pretraining at $L = 8192$ for HF comparison

For the comparison against published HuggingFace checkpoints (Figure 3, right), PatchTST-FM and Chronos-2 are pretrained on SPECTOR at a maximum context length of 8192 so that the model configuration matches the released `ibm-research/patchtst-fm-r1` and `amazon/chronos-2` checkpoints rather than the shared $L = 1024$ protocol above. Both models are evaluated in univariate mode (without cross-learning for Chronos-2) with context 8192. For PatchTST-FM, to match the inference of PatchTST-FM-HF, no autoregressive rollout is used at inference. The architectures are those of Tables 6 and 5; Tables 9 and 10 list the settings that differ.

Table 9: PatchTST-FM pretraining hyperparameters at $L = 8192$ (HF comparison). Architecture and optimizer as in Table 6.

Hyperparameter	**Value**
Context length	8192
Series length / horizon	8192 / 128
Training context	$\mathrm{Unif}[64, 8064]$
Masking	ratio 0.4, contiguous blocks of 8 patches
Prediction tail mask	up to 512 steps (PatchTST-FM default)
Data	SPECTOR on the fly, $L = 8192$ configuration
Batch size	32 per GPU, 8 GPUs, $4\times$ accumulation (1024)
Optimizer	AdamW, lr $3 \cdot 10^{-4}$, $\beta = (0.9, 0.95)$, wd 0.1
LR schedule	5k warmup, cosine to $0.033\times$ peak
Steps	100k
Gradient clipping / precision	1.0 / fp16-mixed

Table 10: Chronos-2 two-stage pretraining to $L = 8192$ (HF comparison). Architecture as in Table 5; stage 2 warm-starts from the stage-1 weights with a fresh optimizer and schedule.

Hyperparameter	**Stage 1**	**Stage 2**
Context length	2048	8192
Max. output patches	16 (256 steps)	64 (1024 steps)
Prediction length	random	random
Time-encoding scale	8192	8192
Data	SPECTOR on the fly, $L = 2048$ configuration	SPECTOR on the fly, $L = 8192$ configuration
Min. context / value-drop prob.	64 / 0.2	64 / 0.2
Initialization	random	stage-1 final checkpoint
Steps	2M	100k
Optimizer	AdamW (fused), lr 10^{-4}	AdamW (fused), lr 10^{-4}
LR schedule	cosine, 5% warmup	cosine, 1% warmup
Batch size	32 per GPU, 8 GPUs (256)	32 per GPU, 8 GPUs (256)

H BENCHMARK AND EVALUATION DETAILS

Benchmark Details. Table tables 11 to 14 give detailed dataset-level descriptions of each benchmark.

Evaluation Protocol. For task-level performance, we follow each benchmark's default metrics: sCRPS for M-Series and GiftEval and SQL for FEV-Bench. We additionally report MASE for point forecasting performance. For benchmark-level performance, we follow Shchur et al. (2025) and aggregate task-level errors into two quantities, defined below. Throughout, a *task* is a (dataset, frequency, horizon) triple; there are 13 tasks in M-Series, 97 in GiftEval, and 100 in FEV-Bench, for a total of 210.

Skill Score. Let $\mathcal{T}$ be the set of tasks in a benchmark, let $e_m(\tau)$ denote the error of model m on task $\tau \in \mathcal{T}$ under the benchmark's default metric, and let $e_{\mathrm{SN}}(\tau)$ denote the error of the Seasonal Naïve baseline on the same task and metric. The Skill Score of m on the benchmark is

$$\mathrm{SS}(m) \;=\; 100 \times \left(1 - \Big(\prod_{\tau \in \mathcal{T}} r_m(\tau)\Big)^{1/|\mathcal{T}|}\right), \qquad r_m(\tau) \;=\; \mathrm{clip}\Big(\frac{e_m(\tau)}{e_{\mathrm{SN}}(\tau)},\, 10^{-2},\, 10^{2}\Big), \quad (66)$$

i.e., one minus the geometric mean of the clipped error ratios relative to Seasonal Naïve, in percent. A score of 0 matches Seasonal Naïve, positive values indicate improvement over it, and the geometric mean makes the aggregate invariant to the scale of individual tasks. Skill Scores are computed w.r.t. sCRPS for M-Series and GiftEval and w.r.t. SQL for FEV-Bench (Table 1), and w.r.t. MASE for point forecasting (Appendix I).

Pairwise Win Rate. For two models a and b evaluated on a common task set $\mathcal{T}$, the pairwise win rate of a over b is the fraction of tasks on which a attains the lower error, with ties counted as half a win:

$$\mathrm{WR}(a \succ b) \;=\; \frac{100}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \Big(\mathbf{1}\{e_a(\tau) < e_b(\tau)\} + \tfrac{1}{2}\,\mathbf{1}\{e_a(\tau) = e_b(\tau)\}\Big). \quad (67)$$

The comparison is paired: both models are evaluated on the same task τ before the indicator is formed. The error $e(\tau)$ is the benchmark's default metric for that task, i.e., sCRPS for M-Series and GiftEval tasks and SQL for FEV-Bench tasks, matching the metrics used for the Skill Score; win rates are then pooled over all 210 tasks of the three benchmarks.

Confidence intervals. All 95% confidence intervals for win rates are percentile bootstrap intervals with $B = 1{,}000$ resamples and a fixed random seed. The resampling unit is the task: in every resample, $|\mathcal{T}|$ tasks are drawn with replacement, the win rate is recomputed on the resampled tasks, and the interval is given by the 2.5th and 97.5th percentiles of the resulting B values. Two variants appear in the paper.

- *Single architecture* (Table 23): a and b are two pretraining priors for the same architecture, $\mathcal{T}$ is the set of 210 tasks, and equation 67 is bootstrapped directly over tasks.
- *Pooled across architectures* (Figure 1(b)): the win rate is pooled over all (task, architecture) pairs for the six architectures in Table 1, i.e., the sum of the indicators in equation 67 over all pairs divided by the number of pairs. The bootstrap remains task-clustered: tasks are resampled with replacement and each drawn task carries the results of all six architectures as a block, so the within-task correlation across architectures is preserved rather than treated as independent evidence.

Table 11: Summary of M-Series forecasting datasets used in our empirical study.

	Frequency	Seasonality	Horizon	Series	Min Length	Max Length
M1	Monthly	12	18	617	48	150
	Quarterly	4	8	203	18	114
	Yearly	1	6	181	15	58
M3	Other	4	8	174	71	104
	Monthly	12	18	1,428	66	144
	Quarterly	4	8	756	24	72
	Yearly	1	6	645	20	47
M4	Hourly	24	48	414	748	1,008
	Daily	1	14	4,227	107	9,933
	Weekly	1	13	359	93	2,610
	Monthly	12	18	48,000	60	2,812
	Quarterly	4	8	24,000	24	874
	Yearly	1	6	23,000	19	841
Tourism	Monthly	12	24	366	91	333
	Quarterly	4	8	427	30	130
	Yearly	1	4	518	11	47

Table 12: Summary of GiftEval forecasting datasets used in our empirical study.

Dataset	Source	Domain	Freq.	# Series	Avg	Min	Max	# Obs
Jena Weather	Autoformer (Wu et al. 2021)	Nature	10T	1	52,704	52,704	52,704	52,704
Jena Weather	Autoformer (Wu et al. 2021)	Nature	H	1	8,784	8,784	8,784	8,784
Jena Weather	Autoformer (Wu et al. 2021)	Nature	D	1	366	366	366	366
BizITObs - Application	AutoMixer (Palaskar et al. 2024)	Web/CloudOps	10S	1	8,834	8,834	8,834	8,834
BizITObs - Service	AutoMixer (Palaskar et al. 2024)	Web/CloudOps	10S	21	8,835	8,835	8,835	185,535
BizITObs - L2C	AutoMixer (Palaskar et al. 2024)	Web/CloudOps	5T	1	31,968	31,968	31,968	31,968
BizITObs - L2C	AutoMixer (Palaskar et al. 2024)	Web/CloudOps	H	1	2,664	2,664	2,664	2,664
Bitbrains - Fast Storage	Grid Workloads Archive (Shen et al. 2015)	Web/CloudOps	5T	1,250	8,640	8,640	8,640	10,800,000
Bitbrains - Fast Storage	Grid Workloads Archive (Shen et al. 2015)	Web/CloudOps	H	1,250	721	721	721	901,250
Bitbrains - rnd	Grid Workloads Archive (Shen et al. 2015)	Web/CloudOps	5T	500	8,640	8,640	8,640	4,320,000
Bitbrains - rnd	Grid Workloads Archive (Shen et al. 2015)	Web/CloudOps	H	500	720	720	720	360,000
Restaurant	Recruit Rest. Comp. (Howard et al. 2017)	Sales	D	807	358	67	478	289,303
ETT1	Informer (Zhou et al. 2020)	Energy	15T	1	69,680	69,680	69,680	69,680
ETT1	Informer (Zhou et al. 2020)	Energy	H	1	17,420	17,420	17,420	17,420
ETT1	Informer (Zhou et al. 2020)	Energy	D	1	725	725	725	725
ETT1	Informer (Zhou et al. 2020)	Energy	W-THU	1	103	103	103	103
ETT2	Informer (Zhou et al. 2020)	Energy	15T	1	69,680	69,680	69,680	69,680
ETT2	Informer (Zhou et al. 2020)	Energy	H	1	17,420	17,420	17,420	17,420
ETT2	Informer (Zhou et al. 2020)	Energy	D	1	725	725	725	725
ETT2	Informer (Zhou et al. 2020)	Energy	W-THU	1	103	103	103	103
Loop Seattle	LibCity (Wang et al. 2023a)	Transport	5T	323	105,120	105,120	105,120	33,953,760
Loop Seattle	LibCity (Wang et al. 2023a)	Transport	H	323	8,760	8,760	8,760	2,829,480
Loop Seattle	LibCity (Wang et al. 2023a)	Transport	D	323	365	365	365	117,895
SZ-Taxi	LibCity (Wang et al. 2023a)	Transport	15T	156	2,976	2,976	2,976	464,256
SZ-Taxi	LibCity (Wang et al. 2023a)	Transport	H	156	744	744	744	116,064
M_DENSE	LibCity (Wang et al. 2023a)	Transport	H	30	17,520	17,520	17,520	525,600
M_DENSE	LibCity (Wang et al. 2023a)	Transport	D	30	730	730	730	21,900
Solar	LSTNet (Lai et al. 2017)	Energy	10T	137	52,560	52,560	52,560	7,200,720
Solar	LSTNet (Lai et al. 2017)	Energy	H	137	8,760	8,760	8,760	1,200,120
Solar	LSTNet (Lai et al. 2017)	Energy	D	137	365	365	365	50,005
Solar	LSTNet (Lai et al. 2017)	Energy	W-FRI	137	52	52	52	7,124
Hierarchical Sales	Mancuso et al. (2020)	Sales	D	118	1,825	1,825	1,825	215,350
Hierarchical Sales	Mancuso et al. (2020)	Sales	W-WED	118	260	260	260	30,680
M4 Yearly	Monash (Godahewa et al. 2021)	Econ/Fin	A-DEC	22,974	37	19	284	845,109
M4 Quarterly	Monash (Godahewa et al. 2021)	Econ/Fin	Q-DEC	24,000	100	24	874	2,406,108
M4 Monthly	Monash (Godahewa et al. 2021)	Econ/Fin	M	48,000	234	60	2,812	11,246,411
M4 Weekly	Monash (Godahewa et al. 2021)	Econ/Fin	W-SUN	359	1,035	93	2,610	371,579
M4 Daily	Monash (Godahewa et al. 2021)	Econ/Fin	D	4,227	2,371	107	9,933	10,023,836
M4 Hourly	Monash (Godahewa et al. 2021)	Econ/Fin	H	414	902	748	1,008	373,372
Hospital	Monash (Godahewa et al. 2021)	Healthcare	M	767	84	84	84	64,428
COVID Deaths	Monash (Godahewa et al. 2021)	Healthcare	D	266	212	212	212	56,392
US Births	Monash (Godahewa et al. 2021)	Healthcare	D	1	7,305	7,305	7,305	7,305
US Births	Monash (Godahewa et al. 2021)	Healthcare	W-TUE	1	1,043	1,043	1,043	1,043
US Births	Monash (Godahewa et al. 2021)	Healthcare	M	1	240	240	240	240
Saugeen	Monash (Godahewa et al. 2021)	Nature	D	1	23,741	23,741	23,741	23,741
Saugeen	Monash (Godahewa et al. 2021)	Nature	W-THU	1	3,391	3,391	3,391	3,391
Saugeen	Monash (Godahewa et al. 2021)	Nature	M	1	780	780	780	780
Temperature Rain	Monash (Godahewa et al. 2021)	Nature	D	32,072	725	725	725	780
KDD Cup 2018	Monash (Godahewa et al. 2021)	Nature	H	270	10,898	9,504	10,920	2,942,364
KDD Cup 2018	Monash (Godahewa et al. 2021)	Nature	D	270	455	396	455	122,791
Car Parts	Monash (Godahewa et al. 2021)	Sales	M	2,674	51	51	51	136,374
Electricity	UCI ML Archive (Trindade 2015)	Energy	15T	370	140,256	140,256	140,256	51,894,720
Electricity	UCI ML Archive (Trindade 2015)	Energy	H	370	35,064	35,064	35,064	12,973,680
Electricity	UCI ML Archive (Trindade 2015)	Energy	D	370	1,461	1,461	1,461	540,570
Electricity	UCI ML Archive (Trindade 2015)	Energy	W-FRI	370	208	208	208	76,960

Table 13: Summary of FEV-Bench forecasting tasks used in our empirical study (Part 1 of 2). We evaluate on 100 tasks spanning univariate, multivariate, and covariate tracks.

Task	Domain	Freq.	H	W	Median len.	# Series	# Targets	# Past cov.	# Known cov.
GIFT-Eval corpus (Aksu et al., 2024)									
BizITObs - L2C	cloud	5T	288	20	31,968	1	7	0	0
BizITObs - L2C	cloud	H	24	20	2,664	1	7	0	0
ETT	energy	15T	96	20	69,680	2	7	0	0
ETT	energy	H	168	20	17,420	2	7	0	0
ETT	energy	D	28	20	724	2	7	0	0
ETT	energy	W	13	5	103	2	7	0	0
Hierarchical Sales	retail	D	28	10	1,825	118	1	0	0
Hierarchical Sales	retail	W	13	10	260	118	1	0	0
Hospital	healthcare	M	12	4	84	767	1	0	0
Jena Weather	nature	10T	144	20	52,704	1	21	0	0
Jena Weather	nature	D	28	11	366	1	21	0	0
Jena Weather	nature	H	24	20	8,784	1	21	0	0
Loop Seattle	mobility	D	28	10	365	323	1	0	0
Loop Seattle	mobility	5T	288	10	105,120	323	1	0	0
Loop Seattle	mobility	H	168	10	8,760	323	1	0	0
M-DENSE	mobility	D	28	10	730	30	1	0	0
M-DENSE	mobility	H	168	10	17,520	30	1	0	0
SZ Taxi	mobility	15T	96	10	2,976	156	1	0	0
SZ Taxi	mobility	H	168	2	744	156	1	0	0
Solar	energy	W	13	1	52	137	1	0	0
Solar	energy	D	28	10	365	137	1	0	0
Macroeconomic datasets									
Australian Tourism	econ	Q	8	2	36	89	1	0	0
FRED-MD - CEE	econ	M	12	20	798	1	3	4	0
FRED-MD - Macro	econ	M	12	20	798	1	51	0	0
FRED-QD - CEE	econ	Q	8	20	266	1	3	4	0
FRED-QD - Macro	econ	Q	8	20	266	1	51	0	0
GVAR	econ	Q	8	10	178	33	6	3	0
US Consumption	econ	M	12	10	792	31	1	0	0
US Consumption	econ	Q	8	10	262	31	1	0	0
US Consumption	econ	Y	5	10	64	31	1	0	0
World CO2 Emissions	econ	Y	5	9	60	191	1	0	0
World Life Expectancy	econ	Y	5	10	74	237	1	0	0
World Tourism	econ	Y	5	2	21	178	1	0	0
Energy generation and consumption									
ENTSO-e Load	energy	15T	96	20	175,292	6	1	0	3
ENTSO-e Load	energy	30T	96	20	87,645	6	1	0	3
ENTSO-e Load	energy	H	168	20	43,822	6	1	0	3
EPF-BE	energy	H	24	20	52,416	1	1	0	2
EPF-DE	energy	H	24	20	52,416	1	1	0	2
EPF-FR	energy	H	24	20	52,416	1	1	0	2
EPF-NP	energy	H	24	20	52,416	1	1	0	2
EPF-PJM	energy	H	24	20	52,416	1	1	0	2
ERCOT	energy	D	28	20	6,452	8	1	0	0
ERCOT	energy	H	168	20	154,872	8	1	0	0
ERCOT	energy	M	12	15	211	8	1	0	0
ERCOT	energy	W	13	20	921	8	1	0	0
GFC12	energy	H	168	10	39,414	11	1	0	1
GFC14	energy	H	168	20	17,520	1	1	0	1
GFC17	energy	H	168	20	17,544	8	1	0	1
Solar with Weather	energy	15T	96	20	198,600	1	1	2	7
Solar with Weather	energy	H	24	20	49,648	1	1	2	7

Table 14: Summary of FEV-Bench forecasting tasks (Part 2 of 2). Continued from Table 13.

Task	Domain	Freq.	H	W	Median len.	# Series	# Targets	# Past cov.	# Known cov.
BOOMLET (Cohen et al., 2025)									
BOOMLET - 1062	cloud	5T	288	20	16,384	1	21	0	0
BOOMLET - 1209	cloud	5T	288	20	16,384	1	53	0	0
BOOMLET - 1225	cloud	T	60	20	16,384	1	49	0	0
BOOMLET - 1230	cloud	5T	288	20	16,384	1	23	0	0
BOOMLET - 1282	cloud	T	60	20	16,384	1	35	0	0
BOOMLET - 1487	cloud	5T	288	20	16,384	1	54	0	0
BOOMLET - 1631	cloud	30T	96	20	10,463	1	40	0	0
BOOMLET - 1676	cloud	30T	96	20	10,463	1	100	0	0
BOOMLET - 1855	cloud	H	24	20	5,231	1	52	0	0
BOOMLET - 1975	cloud	H	24	20	5,231	1	75	0	0
BOOMLET - 2187	cloud	H	24	20	5,231	1	100	0	0
BOOMLET - 285	cloud	T	60	20	16,384	1	75	0	0
BOOMLET - 619	cloud	T	60	20	16,384	1	52	0	0
BOOMLET - 772	cloud	T	60	20	16,384	1	67	0	0
BOOMLET - 963	cloud	T	60	20	16,384	1	28	0	0
Retail and forecasting competitions									
Favorita Store Sales	retail	M	12	2	54	1,579	1	1	1
Favorita Store Sales	retail	W	13	10	240	1,579	1	1	1
Favorita Store Sales	retail	D	28	10	1,688	1,579	1	1	2
Favorita Transactions	retail	M	12	2	54	51	1	1	0
Favorita Transactions	retail	W	13	10	240	51	1	1	0
Favorita Transactions	retail	D	28	10	1,688	51	1	1	1
KDD Cup 2022	energy	D	14	10	243	134	1	9	0
KDD Cup 2022	energy	10T	288	10	35,279	134	1	9	0
KDD Cup 2022	energy	30T	96	10	11,758	134	1	9	0
M5	retail	M	12	1	58	30,490	1	0	8
M5	retail	W	13	1	257	30,490	1	0	8
M5	retail	D	28	1	1,810	30,490	1	0	8
Restaurant	retail	D	28	8	296	817	1	0	0
Rohlik Orders	retail	W	8	5	170	7	1	9	4
Rohlik Orders	retail	D	61	5	1,197	7	1	9	4
Rohlik Sales	retail	W	8	1	150	5,243	1	1	13
Rohlik Sales	retail	D	14	1	1,046	5,390	1	1	13
Rossmann	retail	W	13	8	133	1,115	1	1	4
Rossmann	retail	D	48	10	942	1,115	1	1	5
Walmart	retail	W	39	1	143	2,936	1	0	10
Other sources									
ECDC ILI	healthcare	W	13	10	201	25	1	0	0
Hermes	retail	W	52	1	261	10,000	1	0	1
Hospital Admissions	healthcare	D	28	20	1,731	8	1	0	0
Hospital Admissions	healthcare	W	13	16	246	8	1	0	0
Redset	cloud	5T	288	10	25,920	118	1	0	0
Redset	cloud	15T	96	10	8,640	126	1	0	0
Redset	cloud	H	24	10	2,160	138	1	0	0
UCI Air Quality	nature	H	168	20	9,357	1	4	0	3
UCI Air Quality	nature	D	28	11	389	1	4	0	3
UK COVID - Nation - Cumulative	healthcare	D	28	20	729	4	3	5	0
UK COVID - Nation - Cumulative	healthcare	W	8	4	105	4	3	5	0
UK COVID - Nation - New	healthcare	D	28	20	729	4	3	5	0
UK COVID - Nation - New	healthcare	W	8	4	105	4	3	5	0
UK COVID - UTLA - Cumulative	healthcare	W	13	5	104	214	1	0	0
UK COVID - UTLA - New	healthcare	D	28	10	721	214	1	0	0

I Additional Empirical Results

I.1 Probability Forecasting Performance

Table 15 reports zero-shot probabilistic forecasting performance as the weighted average sCRPS across all datasets within each benchmark, complementing the Skill Scores reported in Table 1.

Table 15: Zero-shot Probabilistic Forecasting Performance (weighted average, sCRPS ↓). M-Series 13 tasks, GiftEval 97, FEV-Bench 100. **Bold** indicates best and underlined indicates second best.

Generator	N-BEATS	MQCNN	Chronos-2	PatchTST-FM	S4D	TiRex
M-Series						
ETS	0.0978	0.0982	**0.0963**	0.0968	0.0978	0.0983
ForecastPFN	0.1035	0.1064	0.1084	0.1033	0.1030	0.1056
KernelSynth	0.1127	0.1095	0.1131	0.1142	0.1107	0.1143
CauKer	0.1055	0.1013	0.1094	0.1047	0.1042	0.1089
SarSim0	0.0970	0.1051	0.0974	0.0948	0.0949	**0.0972**
SPECTOR (Ours)	**0.0954**	**0.0950**	0.0982	**0.0921**	**0.0946**	**0.0972**
GiftEval						
ETS	0.3949	0.4068	0.3988	0.4055	0.3897	0.3973
ForecastPFN	0.3773	0.3957	0.3727	0.3843	0.3914	0.3898
KernelSynth	0.4102	0.4163	0.4262	0.4273	0.4065	0.4212
CauKer	0.3496	0.3485	0.3319	0.3343	0.3338	0.3337
SarSim0	0.3782	0.3781	0.3939	0.3830	0.3743	0.3961
SPECTOR (Ours)	**0.3364**	**0.3411**	**0.3232**	**0.3243**	**0.3256**	**0.3254**
FEV-Bench						
ETS	0.3155	0.3269	0.3200	0.3155	0.3124	0.3259
ForecastPFN	0.3047	0.3184	0.3054	0.3031	0.3065	0.3077
KernelSynth	0.3204	0.3291	0.3208	0.3145	0.3080	0.3282
CauKer	0.2929	0.3003	0.2946	0.2881	0.2852	0.2907
SarSim0	0.2989	0.3084	0.2988	0.2948	0.2910	0.3042
SPECTOR (Ours)	**0.2858**	**0.2923**	**0.2831**	**0.2805**	**0.2799**	**0.2841**

I.2 Point Forecasting Performance

Besides probabilistic forecasting results in Table 1 and 15, here we report point forecasting performance using both average skill score and weighted average w.r.t. MASE across all models and benchmarks in Table 16 and 17, confirming that SPECTOR's advantages extend beyond probabilistic metrics to deterministic accuracy.

I.3 Performance by Frequency

We break down GiftEval performance by sampling frequency to assess whether the gains of SPECTOR are concentrated at particular time scales. Table 18 reports the Skill Score of every architecture–generator combination on the ten GiftEval frequencies. SPECTOR is the best synthetic prior for all six architectures at the 10T, hourly, daily and weekly frequencies, which together account for 60 of the 97 tasks, and for five of six architectures at 5T; the largest margins over the strongest baseline occur at 10T. At the remaining frequencies the picture is mixed: the 10S, quarterly and yearly groups contain one to six tasks each, and the monthly group is dominated by SarSim0 for several architectures, so their scores are correspondingly noisy.

I.4 Performance by Horizon

We break down GiftEval performance by forecast horizon (short, medium, long) to assess how well SPECTOR-pretrained models extrapolate to longer prediction windows, reporting Skill Score (Table 19) and weighted sCRPS (Table 20).

Table 16: Zero-shot Point Forecasting Performance (Skill Score ↑ w.r.t. MASE, relative to Seasonal Naive). M-Series 13 tasks, GiftEval 97, FEV-Bench 100. **Bold** indicates best and underlined indicates second best.

Generator	N-BEATS	MQCNN	Chronos-2	PatchTST-FM	S4D	TiRex
M-Series						
ETS	10.84	2.97	4.58	11.83	11.51	7.93
ForecastPFN	5.18	0.20	−2.63	6.92	5.46	4.29
KernelSynth	2.68	2.59	−2.33	0.86	5.89	−1.63
CauKer	11.05	8.75	4.41	10.14	12.76	8.71
SarSim0	13.42	−2.55	**14.57**	15.11	16.38	13.62
SPECTOR (Ours)	**18.80**	**13.66**	13.77	**20.69**	**20.34**	**16.69**
GiftEval						
ETS	−21.07	−24.89	−9.56	−8.41	−4.84	−23.53
ForecastPFN	−10.08	−23.99	−9.14	−12.64	−8.35	−24.88
KernelSynth	1.96	−7.23	3.47	4.31	7.38	−25.22
CauKer	12.61	5.43	14.47	16.15	17.69	11.05
SarSim0	−1.35	−14.21	2.39	5.71	6.95	−6.21
SPECTOR (Ours)	**16.70**	**8.17**	**20.91**	**21.25**	**20.36**	**13.22**
FEV-Bench						
ETS	9.65	4.73	8.83	11.26	12.39	7.89
ForecastPFN	12.41	3.89	12.13	14.71	13.55	9.45
KernelSynth	12.27	8.75	9.90	13.80	16.13	8.17
CauKer	20.93	17.82	19.69	23.18	24.88	21.25
SarSim0	16.33	7.62	17.87	17.32	18.83	15.58
SPECTOR (Ours)	**23.93**	**20.67**	**24.31**	**27.75**	**27.54**	**24.17**

Table 17: Zero-shot Point Forecasting Performance (weighted average, MASE ↓). M-Series 13 tasks, GiftEval 97, FEV-Bench 100. **Bold** indicates best and underlined indicates second best.

Generator	N-BEATS	MQCNN	Chronos-2	PatchTST-FM	S4D	TiRex
M-Series						
ETS	1.8630	1.8605	1.8076	1.7728	1.8251	**1.8197**
ForecastPFN	1.9522	2.0249	1.9486	1.8497	1.8906	2.0054
KernelSynth	2.1058	2.0627	2.0243	2.2023	2.0513	2.1522
CauKer	1.9810	1.8747	2.0580	2.0224	1.9591	2.1057
SarSim0	1.8201	2.0308	**1.7923**	1.7828	1.7796	1.8797
SPECTOR (Ours)	**1.7476**	**1.7523**	1.7957	**1.6810**	**1.7399**	1.8373
GiftEval						
ETS	1.9372	1.9675	1.7518	1.8015	1.6719	1.9984
ForecastPFN	1.6953	1.7951	1.6037	1.6268	1.6178	1.7963
KernelSynth	1.8095	1.9098	1.7724	1.9166	1.6961	3.3874
CauKer	1.5263	1.5364	1.5134	1.5068	1.4709	1.5600
SarSim0	1.6663	1.8072	1.6242	1.6553	1.5777	1.7019
SPECTOR (Ours)	**1.4377**	**1.4838**	**1.3971**	**1.3808**	**1.3922**	**1.5325**
FEV-Bench						
ETS	1.4749	1.5179	1.4928	1.4621	1.4721	1.5023
ForecastPFN	1.4209	1.5125	1.4004	1.3975	1.3879	1.4336
KernelSynth	1.4962	1.4886	1.5030	1.4855	1.4359	1.5276
CauKer	1.3837	1.4102	1.4134	1.3850	1.3441	1.3794
SarSim0	1.3858	1.4665	1.3781	1.3841	1.3700	1.3874
SPECTOR (Ours)	**1.3529**	**1.3661**	**1.3336**	**1.3110**	**1.3053**	**1.3430**

I.5 Performance by Task Type

We stratify FEV-Bench results by task type (univariate, multivariate, covariate) to examine whether SPECTOR's univariate generation translates to gains on multivariate and covariate forecasting tasks, reporting Skill Score (Table 21) and weighted sCRPS (Table 22).

Table 18: GiftEval performance by sampling frequency (Skill Score ↑ w.r.t. sCRPS, relative to Seasonal Naive). The number of tasks per frequency is given in parentheses; Q and A each contain a single task. **Bold** indicates best and underlined indicates second best.

Generator	10S (6)	5T (12)	10T (6)	15T (12)	H (31)	D (15)	W (8)	M (5)	Q (1)	A (1)
N-BEATS										
ETS	−125.69	24.32	25.93	−54.07	21.99	50.47	17.39	21.05	15.23	−0.76
ForecastPFN	−85.22	18.48	17.60	−13.29	28.60	48.53	26.31	17.11	14.75	−3.67
KernelSynth	9.50	20.84	11.46	20.28	38.65	45.16	22.22	10.73	7.58	−17.75
CauKer	**23.14**	28.85	34.87	17.94	43.16	54.36	31.23	23.45	13.81	−10.71
SarSim0	1.15	26.84	19.12	−16.08	39.69	52.43	31.83	**26.69**	17.10	1.08
SPECTOR (Ours)	15.48	**35.10**	**54.34**	**21.73**	**46.39**	**54.93**	**34.39**	23.42	**21.00**	**2.80**
MQCNN										
ETS	−217.04	22.34	27.80	−60.98	16.63	50.56	4.09	20.94	13.11	3.31
ForecastPFN	−162.45	16.73	5.42	−62.18	20.51	46.87	12.38	17.75	11.68	−8.92
KernelSynth	−49.72	22.42	13.12	**18.49**	36.75	46.80	19.50	14.44	10.96	−16.44
CauKer	**−16.07**	31.14	34.76	13.09	42.12	53.42	28.70	20.94	16.57	−1.58
SarSim0	−65.90	27.51	16.35	−38.39	35.29	51.45	25.81	14.53	13.90	−14.45
SPECTOR (Ours)	−30.02	**34.18**	**38.35**	16.60	**42.33**	**53.81**	**33.12**	**25.95**	**22.19**	**5.34**
Chronos-2										
ETS	−47.43	31.27	27.75	−33.55	29.52	49.96	16.93	17.76	15.06	**8.92**
ForecastPFN	−126.94	24.55	2.88	2.27	36.76	45.37	24.40	16.66	8.09	−3.01
KernelSynth	18.23	15.92	5.66	26.11	42.25	45.70	19.31	8.32	5.92	−5.25
CauKer	24.28	**39.08**	51.56	24.81	48.16	55.69	28.40	18.13	6.43	−15.56
SarSim0	−29.64	20.11	15.58	−6.46	43.11	53.89	35.36	**30.53**	**19.89**	−3.82
SPECTOR (Ours)	**29.39**	39.03	**52.47**	**31.87**	**52.70**	**58.02**	**37.05**	29.52	17.85	1.47
PatchTST-FM										
ETS	−70.84	25.90	29.26	−50.36	27.30	52.17	16.06	21.58	14.57	5.17
ForecastPFN	−315.50	8.89	24.81	−34.94	36.02	49.29	26.96	16.41	12.91	5.01
KernelSynth	17.86	16.95	4.62	22.77	41.45	46.62	18.55	4.26	7.09	−14.62
CauKer	24.82	31.37	44.50	20.38	47.20	56.70	31.14	22.16	15.33	−10.71
SarSim0	**36.45**	20.93	15.97	−21.68	42.50	53.17	32.20	**28.73**	17.84	5.89
SPECTOR (Ours)	26.63	**35.51**	**62.04**	**23.21**	**49.07**	**57.87**	**37.20**	28.43	**22.99**	**11.21**
S4D										
ETS	−49.55	27.40	23.06	−25.05	29.13	52.17	12.84	21.43	15.48	−0.37
ForecastPFN	−155.62	16.57	7.15	−18.27	32.55	47.61	26.60	22.85	15.04	2.64
KernelSynth	3.47	21.69	10.27	24.27	42.87	48.35	25.31	15.24	10.42	−5.96
CauKer	**18.90**	34.41	41.55	24.94	49.15	55.95	35.07	21.50	16.16	−6.62
SarSim0	−1.70	27.69	28.97	−12.07	39.51	54.40	36.34	**30.98**	21.96	1.85
SPECTOR (Ours)	18.89	**36.01**	**55.32**	**25.51**	**49.89**	**56.63**	**38.21**	25.63	**22.96**	**5.45**
TiRex										
ETS	−175.55	27.60	26.96	−50.66	23.54	51.32	11.69	23.64	15.08	**0.85**
ForecastPFN	−213.34	20.38	−11.27	−45.05	25.47	47.32	25.03	23.57	15.99	−11.54
KernelSynth	−44.68	21.22	2.10	13.11	32.08	44.02	26.22	7.41	5.52	−17.61
CauKer	**5.29**	37.74	40.45	22.84	45.19	55.22	37.75	21.94	16.45	−29.74
SarSim0	−96.91	20.33	−4.28	−30.83	31.92	52.81	34.81	27.04	21.03	−6.65
SPECTOR (Ours)	−24.60	**40.59**	**47.11**	**24.20**	**49.10**	**56.29**	**40.63**	**31.56**	**23.65**	−7.86

Table 19: GiftEval performance by forecast horizon (Skill Score ↑ w.r.t. sCRPS, relative to Seasonal Naive). **Bold** indicates best and underlined indicates second best.

Generator	N-BEATS	MQCNN	Chronos-2	PatchTST-FM	S4D	TiRex
Short horizon (55 tasks)						
ETS	30.15	26.53	29.34	29.63	31.65	28.60
ForecastPFN	31.77	23.73	30.05	33.51	31.41	29.55
KernelSynth	31.55	30.44	32.39	31.93	34.75	29.44
CauKer	40.63	38.93	42.79	43.95	44.01	42.55
SarSim0	36.07	32.44	37.65	38.54	39.11	36.57
SPECTOR (Ours)	**42.64**	**40.65**	**46.50**	**46.93**	**46.04**	**44.99**
Medium horizon (21 tasks)						
ETS	−7.73	−17.15	9.87	3.58	8.22	−4.74
ForecastPFN	2.73	−17.03	8.83	−2.61	−0.61	−13.00
KernelSynth	25.80	21.49	28.37	27.30	27.45	15.83
CauKer	30.40	**27.35**	39.61	33.81	35.94	34.35
SarSim0	22.78	12.92	19.88	24.06	22.49	3.13
SPECTOR (Ours)	**36.55**	27.30	**45.01**	**41.00**	**38.99**	**37.25**
Long horizon (21 tasks)						
ETS	−9.97	−17.51	14.59	5.76	8.19	−5.70
ForecastPFN	3.36	−9.86	16.52	−18.66	1.09	−18.01
KernelSynth	26.22	21.28	28.58	27.48	29.67	13.60
CauKer	31.62	26.47	38.63	33.74	37.20	33.45
SarSim0	20.84	4.86	22.53	22.20	20.92	−10.86
SPECTOR (Ours)	**37.79**	**27.70**	**43.61**	**37.38**	**38.67**	**34.94**

Table 20: GiftEval performance by forecast horizon (weighted average, sCRPS ↓). **Bold** indicates best and underlined indicates second best.

Generator	N-BEATS	MQCNN	Chronos-2	PatchTST-FM	S4D	TiRex
Short horizon (55 tasks)						
ETS	0.3764	0.3848	0.3890	0.3854	0.3767	0.3837
ForecastPFN	0.3645	0.3746	0.3689	0.3677	0.3685	0.3681
KernelSynth	0.4139	0.4200	0.4237	0.4272	0.4070	0.4223
CauKer	0.3487	0.3480	0.3390	0.3341	0.3350	0.3372
SarSim0	0.3698	0.3661	0.3730	0.3645	0.3624	0.3621
SPECTOR (Ours)	**0.3392**	**0.3403**	**0.3287**	**0.3261**	**0.3268**	**0.3294**
Medium horizon (21 tasks)						
ETS	0.4897	0.5326	0.4353	0.4895	0.4391	0.4691
ForecastPFN	0.4261	0.4944	0.3798	0.3941	0.4577	0.5080
KernelSynth	0.3766	0.3862	0.4053	0.3881	0.3890	0.3955
CauKer	0.3433	0.3326	0.2747	0.3150	0.3142	0.3003
SarSim0	0.4105	0.4233	0.4666	0.4576	0.4166	0.5188
SPECTOR (Ours)	**0.3054**	**0.3290**	**0.2740**	**0.2919**	**0.2978**	**0.2858**
Long horizon (21 tasks)						
ETS	0.5299	0.5519	0.4868	0.5726	0.5039	0.4939
ForecastPFN	0.4904	0.5603	0.4144	0.5905	0.6153	0.5396
KernelSynth	0.3999	0.4013	0.4817	0.4742	0.4196	0.4355
CauKer	0.3681	0.3726	**0.3052**	0.3588	**0.3399**	0.3257
SarSim0	0.4519	0.4844	0.5836	0.5401	0.4822	0.7021
SPECTOR (Ours)	**0.3354**	**0.3642**	0.3062	**0.3380**	0.3415	**0.3177**

Table 21: FEV-Bench performance by task type (Skill Score ↑ w.r.t. SQL, relative to Seasonal Naive). **Bold** indicates best and underlined indicates second best.

Generator	N-BEATS	MQCNN	Chronos-2	PatchTST-FM	S4D	TiRex
Univariate (32 tasks)						
ETS	18.98	13.96	16.95	18.65	19.86	15.77
ForecastPFN	19.17	9.72	20.96	23.44	22.75	17.95
KernelSynth	19.31	14.76	18.13	20.21	23.46	18.71
CauKer	26.11	23.76	26.07	26.88	29.87	27.35
SarSim0	23.01	13.22	24.48	23.48	23.74	24.15
SPECTOR (Ours)	**28.03**	**25.49**	**30.30**	**31.29**	**31.71**	**29.90**
Multivariate (26 tasks)						
ETS	37.26	35.28	39.12	39.13	39.32	36.30
ForecastPFN	39.42	36.03	41.53	42.73	41.25	39.50
KernelSynth	34.90	33.39	35.44	39.63	41.93	29.30
CauKer	47.09	45.45	50.17	50.67	50.74	47.99
SarSim0	42.66	38.33	45.98	46.19	45.58	41.02
SPECTOR (Ours)	**50.11**	**47.56**	**52.44**	**53.06**	**52.37**	**51.23**
Covariate (42 tasks)						
ETS	19.44	14.54	17.97	20.19	21.31	17.93
ForecastPFN	22.41	14.42	20.53	22.86	22.39	20.49
KernelSynth	24.63	22.56	22.84	26.96	27.90	20.94
CauKer	31.26	28.05	27.39	33.55	34.90	31.40
SarSim0	25.38	19.20	26.96	26.48	28.95	24.93
SPECTOR (Ours)	**33.16**	**30.81**	**32.21**	**36.99**	**37.17**	**33.06**

Table 22: FEV-Bench performance by task type (weighted average, sCRPS ↓). **Bold** indicates best and underlined indicates second best.

Generator	N-BEATS	MQCNN	Chronos-2	PatchTST-FM	S4D	TiRex
Univariate (32 tasks)						
ETS	0.2079	0.2161	0.2107	0.2088	0.2048	0.2125
ForecastPFN	0.2022	0.2127	0.1972	0.1972	0.1936	0.1994
KernelSynth	0.2155	0.2268	0.2184	0.2099	0.2020	0.2209
CauKer	0.1904	0.1935	0.1894	0.1868	**0.1801**	0.1868
SarSim0	0.1975	0.2047	0.1961	0.1941	0.1896	0.2061
SPECTOR (Ours)	**0.1861**	**0.1891**	**0.1827**	**0.1831**	0.1807	**0.1866**
Multivariate (26 tasks)						
ETS	0.4718	0.4930	0.4568	0.4557	0.4513	0.4839
ForecastPFN	0.4402	0.4889	0.4485	0.4230	0.4867	0.4483
KernelSynth	0.4701	0.5080	0.4849	0.4429	0.4233	0.5134
CauKer	0.3834	0.4015	0.3457	0.3491	0.3533	0.3747
SarSim0	0.4316	0.4789	0.4008	0.3874	0.3951	0.4405
SPECTOR (Ours)	**0.3610**	**0.3913**	**0.3366**	**0.3319**	**0.3412**	**0.3461**
Covariate (42 tasks)						
ETS	0.3211	0.3321	0.3283	0.3228	0.3201	0.3326
ForecastPFN	0.3117	0.3218	0.3127	0.3127	0.3105	0.3153
KernelSynth	0.3262	0.3309	0.3244	0.3227	0.3182	0.3303
CauKer	0.3052	0.3122	0.3122	0.3036	0.3007	0.3040
SarSim0	0.3059	0.3114	0.3097	0.3064	0.3015	0.3100
SPECTOR (Ours)	**0.2992**	**0.3038**	**0.2993**	**0.2963**	**0.2949**	**0.2987**

Table 23: Comparison of SPECTOR and baseline generators using SPECTOR Augmentation pipeline. Pairwise win rate over 210 tasks in 3 benchmarks with 95% CI.

Comparison	N	**Win rate (%)**	**95% CI**
SPECTOR vs KernelSynth + Aug.	210	70.0	[63.8, 75.7]
SPECTOR vs CauKer + Aug.	210	71.9	[65.7, 78.1]

I.6 Comparison of SPECTOR with baseline generators with augmentation

Table 24 compares performance of SPECTOR with performance of all baseline generators in composition with SPECTOR augmentation.

Table 24: Comparison of SPECTOR and baseline generators using the same augmentation pipeline. **Bold** indicates best and underlined indicates second best.

Generator	**M-Series**	**GiftEval**	**FEV-Bench**
ETS	13.85	30.42	31.46
ForecastPFN	18.08	32.46	32.93
KernelSynth	16.12	41.97	38.08
CauKer	19.07	41.83	37.75
SarSim0	17.87	36.93	35.77
SPECTOR (Ours)	**20.96**	**43.72**	**39.99**

I.7 Mixture Ablation

Figure 6 reports the leave-one-out mixture ablation discussed in Section 3. PatchTST-FM is trained on a uniform mixture of all six synthetic generators (ETS, ForecastPFN, SarSim0, KernelSynth, CauKer, and SPECTOR), drawing each batch from one uniformly selected prior, and we then remove each of the three strongest standalone generators in turn. The full mixture does not improve over standalone SPECTOR on M-Series or FEV-Bench and only marginally on GiftEval, and removing SPECTOR causes by far the largest degradation among the leave-one-out variants.

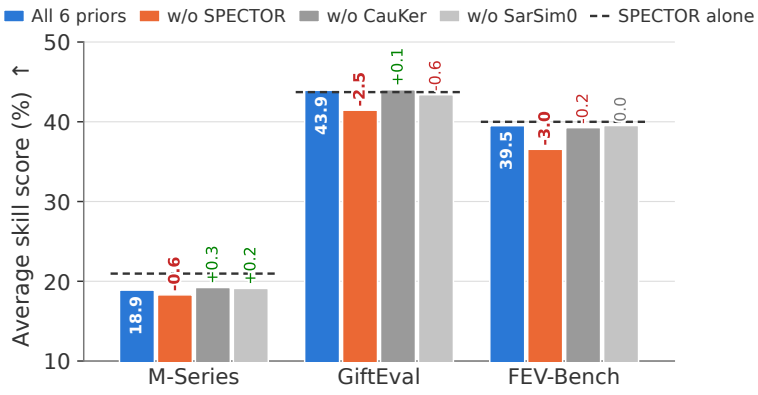
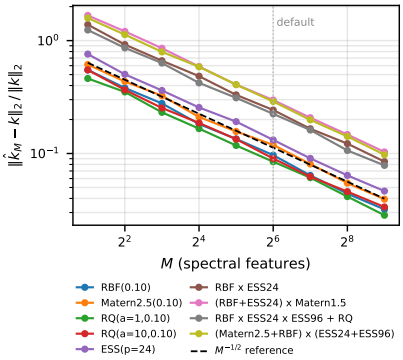


Figure 6: **Mixture ablation.** Leave-one-out ablation of the 6-prior mixture for PatchTST-FM (Skill Score %) on M-Series, GiftEval, and FEV-Bench. Removing SPECTOR costs far more than removing any other prior, and the full mixture does not outperform standalone SPECTOR.

I.8 Impact of Number of Spectral Features

The only approximation in the SPECTOR pipeline enters through the number of spectral features M in Corollary D.2. Theorem 2.3 draws every frequency ω_m exactly from the composite spectral

law, and the time-domain realization $f_M(t) = \sqrt{2k(0)/M}\sum_{m=1}^{M}\cos(\omega_m t + b_m)$ then represents that law by M sampled frequencies. Each realization is a random finite-rank kernel draw: the kernel it induces, $\hat{k}_M(\tau) = \frac{k(0)}{M}\sum_m \cos(\omega_m \tau)$, is unbiased for the target $k(\tau)$ for every finite M (Corollary D.2(i)), while its Monte Carlo fluctuation and the deviation of the finite-dimensional laws from the Gaussian limit shrink with M (Corollary D.2(ii)–(iii)). Generation cost grows linearly in M (Corollary D.2(iv)), so M trades fidelity to the exact GP prior against throughput. All experiments in the main text use $M = 64$. Figure 7 measures the approximation directly and Table 25 its downstream effect. Zero-shot skill is remarkably insensitive to M: across a $16\times$ range ($M = 16$ to 256) the scores vary by at most 0.9 pp on any benchmark, with no monotone trend, so the default $M = 64$ sits comfortably on the plateau while keeping generation cost low.



M	M-Series	GiftEval	FEV-Bench
16	20.71	43.22	39.46
32	**21.44**	**44.10**	39.68
64	20.96	43.72	39.99
128	21.13	43.92	**40.08**
256	20.89	43.77	40.00

Table 25: Zero-shot skill score (%, ↑) of PatchTST-FM pretrained on SPECTOR with different numbers of spectral features M (all other settings as in the shaded row of Table 26; $M = 64$ is the default used for all main results 1).

Figure 7: **Effect of the number of spectral features M.** Relative L_2 error of the induced kernel $\hat{k}_M$ against the analytical composite kernel k on lags $\tau < T/4$ ($T = 1024$, mean over 200 draws) for nine stationary leaves and composites of the SPECTOR bank; the dashed line is the $M^{-1/2}$ Monte Carlo rate and the dotted line marks the default $M = 64$. The per-draw fluctuation at $M = 64$ is 5–15%, while the covariance is exact in expectation at every M.

I.9 Ablation Studies

Table 26: Ablations of different components in the full SPECTOR pipeline, including sampling, kernel bank, and augmentation. All ablations use PatchTST-FM and report skill score % ↑).

Method					M-Series	GiftEval			FEV-Bench		
Data Budget	Sampling	Aug.	Exp. Bank	L	(13)	Short (55)	Medium (21)	Long (21)	Univariate (32)	Multivariate (26)	Covariate (42)
10M	Cholesky	×	×	1024	1.94	31.93	27.30	27.48	20.21	39.63	26.96
10M	SPECTOR	×	×	1024	4.43	38.99	32.16	32.43	24.91	47.75	28.26
OTF	SPECTOR	×	×	1024	2.31	39.23	31.62	33.11	24.05	47.50	28.50
OTF	SPECTOR	×	✓	1024	20.63	43.51	30.69	27.31	29.90	49.19	35.32
OTF	SPECTOR	✓	×	1024	14.76	45.38	39.81	36.69	28.46	52.01	34.87
OTF	SPECTOR	✓	✓	1024	**20.96**	46.93	41.00	37.38	31.29	53.06	36.99
OTF	SPECTOR	✓	✓	2048	20.53	48.43	42.63	38.95	31.62	54.52	38.03
OTF	SPECTOR	✓	✓	4096	20.75	48.40	43.27	40.85	31.89	54.93	38.53
OTF	SPECTOR	✓	✓	8192	19.92	**48.76**	**52.20**	**52.61**	**33.12**	**56.56**	**38.95**

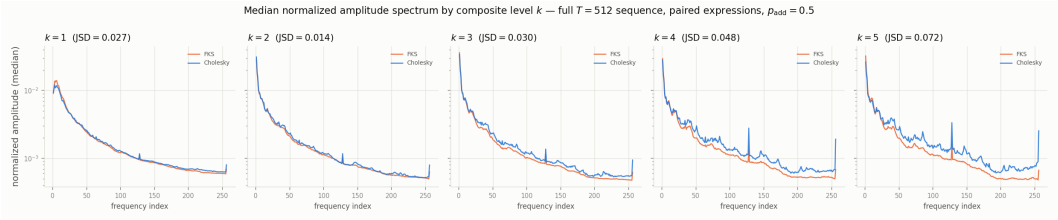


Figure 8: As composite depth increases, Cholesky sampling yields a higher power of high-frequency components while SPECTOR does not.

J Comparison of Covariance-based Sampling and SPECTOR Sampling

In this section, we study how covariance-based sampling (which KernelSynth uses) and SPECTOR sampling differs.

Figure 8 compares the average power spectra of the two samplers as a function of the depth of the composite kernel. Exact draws accumulate mid- and high-frequency power as the composition deepens, whereas SPECTOR draws do not: for the stationary kernels in the bank the finite set of M sampled frequencies constrains the spectral complexity of every composite, and the effect compounds with depth, giving up to $1.5\times$ lower mid- and high-frequency energy while the low-frequency structure of the exact prior is retained.

K Additional Synthetic Evaluation and Qualitative Analysis

Section 3 shows that pretraining on SPECTOR outperforms pretraining on baseline synthetic priors on real-world benchmarks, but not *what* the prior changes in the resulting model. This appendix section studies the six PatchTST-FM variants in Table 1 and ablated variants of SPECTOR on synthetic inputs to gain a better understanding of what inductive biases SPECTOR provides. See Yu et al. (2026) for similar analyses and a discussion of why these analyses are important in revealing the inductive biases of each model.

SPECTOR generalizes to OOD periods absent from the kernel bank, including periods longer than the training forecast window. We define a sinusoid as in-distribution (ID) if its period lies on the periodic-kernel grid shared by the legacy and expanded banks, and OOD otherwise. We evaluate 100 unit-amplitude sinusoids with random phase at five ID periods (4, 12, 48, 96, 365) and four OOD periods (16, 57, 128, 133), using context 896 and horizon 128. Both Table 27 and Figure 9 show that all kernel-based priors perform well on ID periods, whereas SPECTOR is the only prior that generalizes reliably to OOD periods. This advantage is concentrated at OOD periods at or beyond the training horizon: at periods 16 and 57, SPECTOR is competitive with the best prior, while at 128 and 133 it maintains accurate phase extrapolation as KernelSynth, CauKer, and SarSim0 revert toward the mean and ForecastPFN and ETS extrapolate the slow oscillation as a trend. Because SPECTOR and KernelSynth use the same periodic-kernel grid, this behavior reflects improved OOD generalization rather than broader kernel coverage.

Expanded bank with more non-stationary kernels yields significant improvement in regime / structural break scenarios. Table 28 ablates the SPECTOR pipeline on 19 parametric synthetic tasks. The 13 smooth trend, seasonality, and noise tasks show little difference across variants, while **the gains concentrate on regime and mixed scenarios**. Relative to the legacy-bank SPECTOR model, expanding the bank reduces loss by 68% on mean shifts, 66% on GARCH variance, and 62% on the full mixture. Augmentation further reduces the overall loss, from 0.349 to 0.165 (53%), primarily by improving the remaining regime failures. Thus, expanding the kernel bank is the main source of improvement on structural changes, while augmentation provides an additional refinement.

Table 27: **Comparison of PatchTST-FM variants pretrained on different generators on synthetic in-distribution and out-of-distribution periodicity.** SPECTOR outperforms KernelSynth and CauKer in tasks with OOD periodic patterns.

	sCRPS		MASE	
Pretraining prior	ID	OOD	ID	OOD
ETS	0.807	1.634	19.19	30.63
ForecastPFN	0.336	0.980	11.96	22.45
SarSim0	0.148	0.292	2.83	6.52
KernelSynth	**0.011**	<u>0.175</u>	<u>0.32</u>	<u>4.26</u>
CauKer	<u>0.023</u>	0.183	0.41	4.47
SPECTOR (Ours)	**0.011**	**0.051**	**0.27**	**0.87**

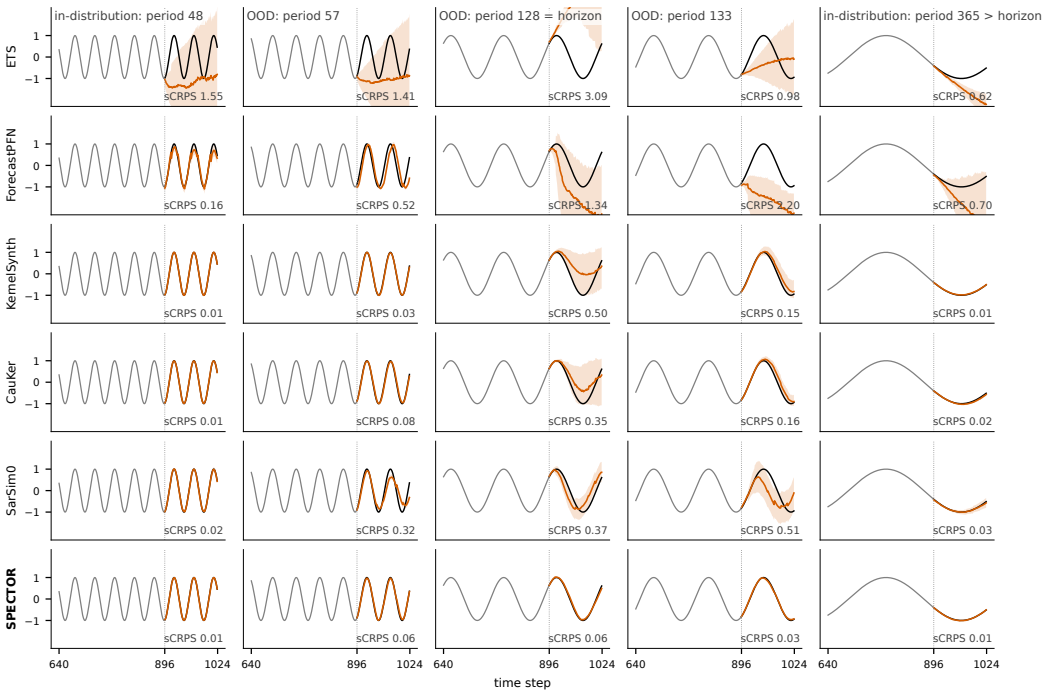


Figure 9: Forecasts on in-distribution vs. out-of-distribution periodicity. Median forecast (orange) with 10–90% band for the six PatchTST-FM checkpoints (rows). Only SPECTOR generalizes to OOD periods beyond training horizon.

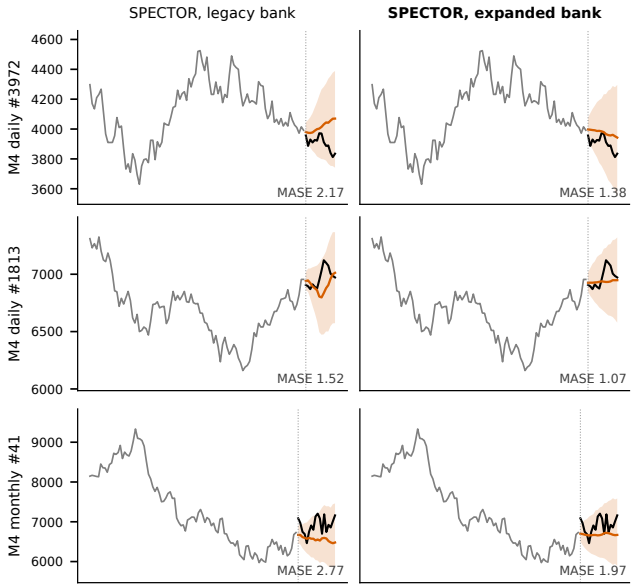


Figure 10: Anecdotes from M-Series (M4): forecast by PatchTST-FM pretrained on SPECTOR with the legacy bank (left) and with the expanded bank (right), both with augmentation.

Table 28: Ablation of the SPECTOR pipeline on 19 parametric synthetic tasks (scaled pinball loss, lower is better; PatchTST-FM). Percentages in parentheses denote relative loss change from SPECTOR with the legacy bank. Green and red indicate improvements and regressions greater than 10%, respectively.

Scenario	SPECTOR w/ legacy bank	SPECTOR w/ expanded bank	SPECTOR w/ expanded bank + aug. (Ours)
Trend isolation			
Linear	0.027	0.025 (-7%)	0.026 (-4%)
Quadratic	0.023	0.022 (-4%)	0.027 (+17%)
Sigmoid	0.007	0.007 (0%)	0.007 (0%)
Group mean	0.019	0.018 (-5%)	0.020 (+5%)
Seasonality isolation			
Period 7	0.007	0.007 (0%)	0.007 (0%)
Period 24	0.007	0.006 (-14%)	0.006 (-14%)
Period 52	0.007	0.007 (0%)	0.007 (0%)
Double (7+24)	0.010	0.011 (+10%)	0.012 (+20%)
Atypical (16)	0.007	0.007 (0%)	0.008 (+14%)
Atypical (100)	0.008	0.007 (-13%)	0.009 (+13%)
Group mean	0.008	0.007 (-13%)	0.008 (0%)
Noise isolation			
Gaussian	0.040	0.038 (-5%)	0.038 (-5%)
Poisson	0.038	0.035 (-8%)	0.036 (-5%)
Lognormal	0.014	0.015 (+7%)	0.015 (+7%)
Gamma	0.013	0.013 (0%)	0.013 (0%)
Group mean	0.026	0.025 (-4%)	0.025 (-4%)
Regime / structural			
Mean shift	0.115	0.037 (-68%)	0.018 (-84%)
Variance (GARCH)	3.436	1.169 (-66%)	0.945 (-72%)
Zero-inflated	1.055	0.996 (-6%)	0.821 (-22%)
Group mean	1.535	0.734 (-52%)	0.594 (-61%)
Mixed compositions			
Trend+Season+Noise	0.038	0.039 (+3%)	0.039 (+3%)
Full mix	0.160	0.061 (-62%)	0.072 (-55%)
Zero+Regime	1.614	1.377 (-15%)	1.039 (-36%)
Group mean	0.604	0.492 (-19%)	0.383 (-37%)
Mean (19 tasks)	**0.349**	**0.204 (-42%)**	**0.165 (-53%)**

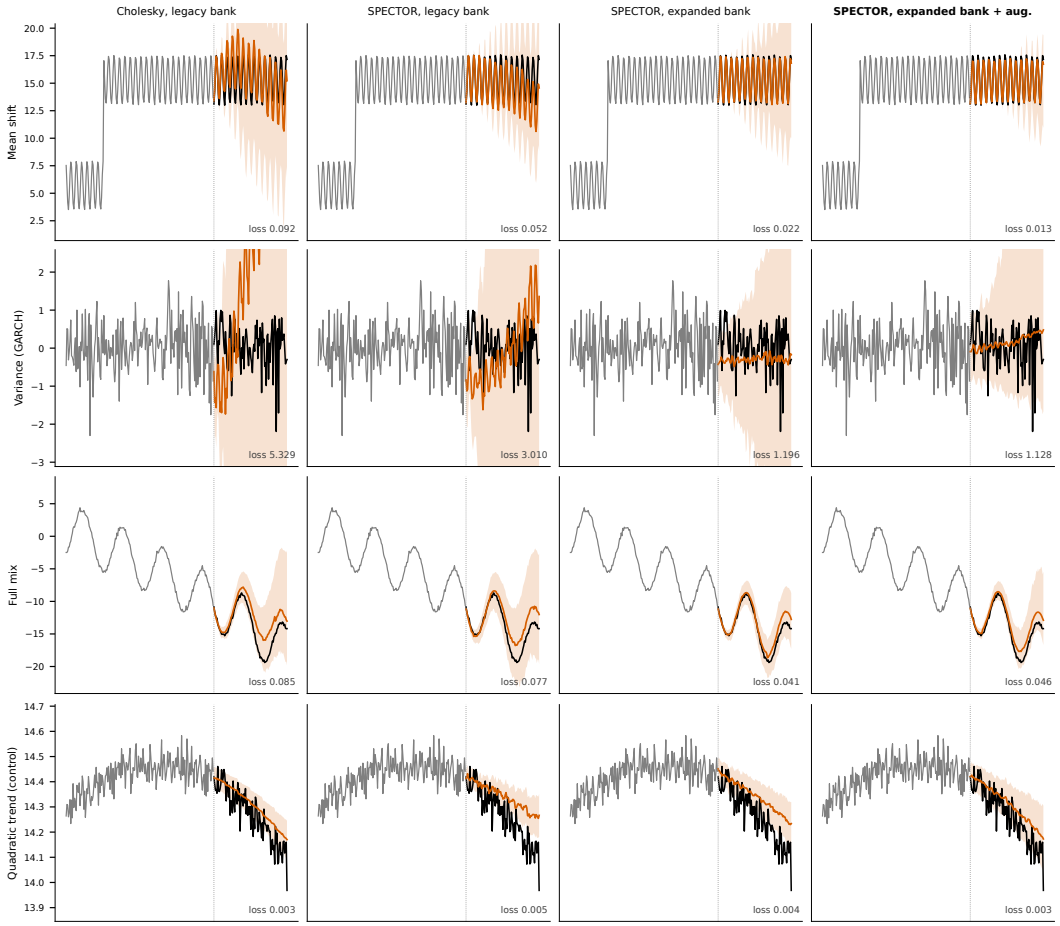


Figure 11: Forecast anecdotes for the pipeline ablation of Table 28 on the mean-shift, GARCH-variance and full-mixture tasks and on the quadratic-trend control (rows), for the four PatchTST-FM checkpoints (columns). For each task the plotted series is the one with the median loss difference between the two bank variants among the 200 scored series. Grey: last 192 context steps; black: truth over the 96-step horizon; colour: median with 10–90% band; scaled pinball loss per panel.