LEARNING FROM ZEROS: REGIME-AWARE SPARSE FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse time series, where a large fraction of observations are exactly zero, are common in retail and supply chain forecasting. Conventional methods for this multi-horizon panel setting address only one facet of the challenge at a time, such as smoothing zeros away, decomposing occurrence and magnitude, or routing series by a fixed sparsity threshold. We argue that the temporal arrangement of zeros carries information about an underlying state and should be modeled as a continuous latent regime shared across the forecast horizon. We show theoretically that marginal objectives cannot identify how sparse events are jointly organized across a forecast window, that a shared latent regime can induce this dependence, and that the independent per-horizon regimes underlying common architectures cannot express it regardless of training. We propose a regime-aware framework built on this insight, combining a sparsity-aware encoder with a context-conditioned latent regime to produce a joint predictive distribution over Negative Binomial outputs. Across a large-scale proprietary e-commerce dataset and three public benchmarks, our method achieves consistent improvements over existing approaches, with joint-level calibration metrics confirming our theoretical predictions.

1 INTRODUCTION

Demand forecasting is a long-standing and widely studied subfield of time-series forecasting, underpinning retail and related supply chain operations (Fildes et al., 2022). The problem we study here is not a conventional one: conventional methods implicitly assume historical observations carry information to be extracted, but in our setting a large fraction of targets in a multi-target forecasting panel are exactly zero, often exceeding half. When most of a series is the same constant, these observations carry almost no information for extrapolation; this is not data scarcity but data that has gone silent. This is common, and forecast accuracy translates directly into inventory cost and stockout loss, so small systematic biases compound into substantial operational cost at scale.

Existing methods each address one facet of this problem: smoothing zeros away as noise (Hyndman et al., 2002), decomposing occurrence from magnitude (Croston, 1972), modeling zero-inflation explicitly (Lambert, 1992), routing series by a sparsity threshold (Wolff et al., 2025), or transferring priors from pretrained foundation models (Ansari et al., 2025). We instead approach it from the perspective of uncertainty: this problem involves two distinct sources that none of these methods distinguish. The usual variation of a value around its mean is aleatoric, irreducible regardless of how much data is collected. But uncertainty about which state the underlying process is currently in, for instance whether a series is out of stock or simply between infrequent purchases, is epistemic, resolvable in principle with more information.

We propose a regime-aware framework for joint probabilistic forecasting, built on this insight. At its core is a latent regime, shared across the forecast horizon and inferred from the temporal arrangement of zeros. Combined with a sparsity-aware encoder, this regime produces a joint predictive distribution over Negative Binomial outputs. On a large-scale real-world proprietary e-commerce dataset and three public benchmarks, Car Parts (Godahewa et al., 2020), Online Retail II (Chen, 2012), and M5 (Makridakis et al., 2022), our method consistently outperforms more than ten baselines. This work begins with an operational problem, builds the theory needed to address it, and returns to that same setting for validation.

Our contributions are: (1) we argue that zeros carry information about the underlying state, and that it should be modeled as a continuous latent regime shared across the forecast horizon, rather than smoothed or thresholded away; (2) we show marginal objectives cannot identify cross-horizon dependence, that a shared regime induces it, and that independent per-horizon regimes cannot express

Code is available at `https://anonymous.4open.science/r/TS_ICLR-C123/`.

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

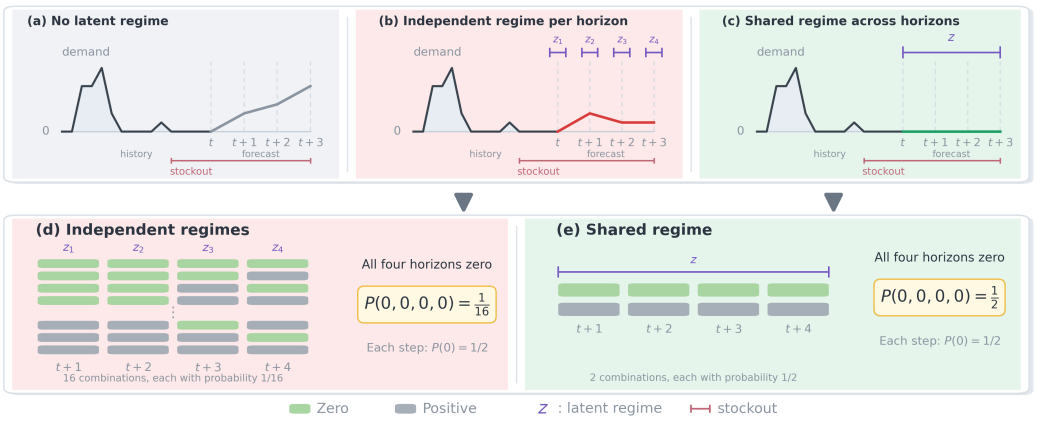

Figure 1: Illustration of why a shared latent regime matters for sparse forecasting. (a) Without any regime, the forecast drifts away. (b) An independent regime per horizon still fails to represent this: the joint probability of the stockout continuing is diluted across 16 equally-likely combinations. (c) A single shared regime concentrates probability on the 2 combinations that are actually coherent. Marginals are identical across (b) and (c); only the joint distribution differs.

it regardless of training; and (3) we provide large-scale validation on a real-world proprietary e-commerce demand forecasting system in addition to public benchmarks, with joint-level calibration metrics providing empirical support for these predictions.

2 Related Work

General-Purpose Time Series Forecasting: General-purpose forecasters fall into a few families. Classical methods fit a model per series and select its configuration automatically (AutoETS (Hyndman et al., 2002), DynamicOptimizedTheta (Fiorucci et al., 2016), AutoARIMA (Hyndman & Khandakar, 2008)), NPTS (Alexandrov et al., 2020) resamples past observations directly, and tree-based regression casts forecasting as supervised learning on lag features (RecursiveTabular (Shchur et al., 2023)). Neural models include recurrent and Transformer-based architectures (DeepAR (Salinas et al., 2020), TFT (Lim et al., 2021), PatchTST (Nie et al., 2022), Autoformer (Wu et al., 2021)) and convolutional or linear/MLP-based ones (WaveNet (Van Den Oord et al., 2016), DLinear (Zeng et al., 2023), TiDE (Das et al., 2023), SimpleFeedForward (Alexandrov et al., 2020)). More recently, pretrained foundation models (Chronos-Bolt and Chronos2 (Ansari et al., 2025), building on Chronos (Ansari et al., 2024)) can be applied zero-shot or fine-tuned on a target domain. Most of these models output a marginal distribution or a set of quantiles for each horizon, while autoregressive ones (DeepAR, WaveNet, Chronos) can also sample joint trajectories, whose cross-horizon dependence comes from feedback rather than an explicit regime. However, these are general-purpose models, and none gives zeros a special role in its design. See Appendix A for the modeling logic behind each family.

Sparse / Intermittent Demand Forecasting: Sparse or intermittent demand has been addressed at several stages of the forecasting pipeline. Temporal aggregation forecasts on a coarser series and disaggregates the result, reducing intermittence at a single scale (ADIDA (Nikolopoulos et al., 2011)) or combining forecasts from several scales (IMAPA (Petropoulos & Kourentzes, 2015)). Croston's method (Croston, 1972) smooths demand sizes and the intervals between demands separately; SBA (Syntetos & Boylan, 2005) reduces its bias, and TSB (Teunter et al., 2011) replaces the interval with an occurrence probability updated in every period, so forecasts decay while a series stays inactive.

At the output-distribution stage, zero-inflated models (Lambert, 1992) add a latent switch that separates structural zeros from sampling zeros, and the Tweedie distribution (Tweedie et al., 1984) places an atom at zero through a compound Poisson structure. At the architectural level, SPADE-S (Wolff et al., 2025) sends extremely sparse series to a dedicated sparse branch instead of the main network, and copula-based models such as the low-rank Gaussian copula process (Salinas et al., 2019) capture dependence between series. In multi-horizon forecasters built on zero-inflated or Tweedie heads, occurrence is typically decided per horizon: independently at each horizon, or coupled only through

autoregressive decoding. To our knowledge, none of the methods above infers a latent regime from the arrangement of zeros and shares it across the forecast horizon.

3 METHOD

3.1 PROBLEM FORMULATION

We consider joint multi-step probabilistic forecasting for a panel of N non-negative integer-valued (count) time series. For series i at forecast origin t, the model observes a history $\mathcal{H}_{i,t} = \{(y_{i,\tau}, x_{i,\tau})\}_{\tau=t-L+1}^{t}$ of length L and, where available, future-known covariates $\mathcal{F}_{i,t} = \{c_{i,t+h}\}_{h=1}^{H}$ over a horizon of H steps (e.g., calendar features and known holidays). We write $I_{i,t}$ for all information available when forecasting series i at origin t (the index i is dropped when clear from context), which subsumes $\mathcal{H}_{i,t}$, $\mathcal{F}_{i,t}$, and any auxiliary or relational signals introduced later. We target series with substantial probability mass at zero, measured by the empirical zero rate $\hat{\pi}_{i,t}^{(0)} = L^{-1} \sum_{\tau=t-L+1}^{t} \mathbf{1}\{y_{i,\tau} = 0\}$ over the context window. Rather than a point estimate of the future trajectory $Y_{i,t+1:t+H}$, our objective is its conditional joint distribution,

$$p_\theta(Y_{i,t+1:t+H} \mid I_{i,t}),$$

from which horizon-wise marginals and quantiles are recovered.

3.2 ARCHITECTURE OVERVIEW

The framework is organized around a single data flow (Figure 2) with three main components. A **sparsity-aware historical encoder** (Section 3.3) maps the history $\mathcal{H}_{i,t}$ to a sequence representation E_{enc} and a summary state e_t. A **context-conditioned latent regime** z (Section 3.5) is inferred from e_t and the future-known context $\mathcal{F}_{i,t}$, and a single realization of z is shared across all H horizons. A **target decoder** (Section 3.4) combines E_{enc} and $\mathcal{F}_{i,t}$ into horizon-specific representations, which are combined with z to parameterize a **joint predictive distribution** (Section 3.6) over the forecast window. Two optional components enrich this core when the corresponding information is available. A **relational signal** $g_{i,t}$ (Section 3.7) brings in information from related series. An **auxiliary decoder** (Section 3.4), built like the target decoder but with its own parameters, forecasts a causally relevant process that is observed historically but not realized at forecast time, supplying a forward-looking signal $\hat{A}$. Both are appended to the future context read by the latent regime and the target decoder.

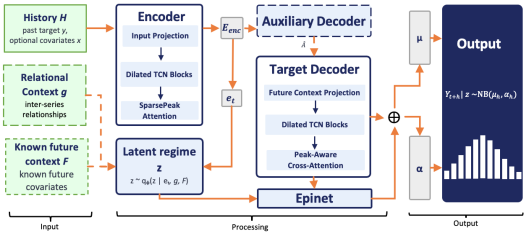

Figure 2: Model overview. Dashed borders denote optional components.

3.3 SPARSITY-AWARE HISTORICAL ENCODER

The encoder maps the history $\mathcal{H}_{i,t}$ to a sequence representation E_{enc} and a summary state e_t. Input features are projected into a hidden space and passed through a stack of dilated causal convolutions (Bai et al., 2018), followed by a self-attention layer. We use dilated convolutions because zeros in sparse series typically occur in consecutive runs of varying length, and increasing dilation rates provide receptive fields that match runs of different durations.

In sparse series most positions are zero, so standard self-attention, which weighs positions purely by query-key similarity, can spread most of its weight over zeros, which say little about magnitude, and dilute the few positive observations. We therefore add two fixed biases to the attention score of key position j, $s_j = q^\top k_j / \sqrt{d} - \lambda\, \mathbf{1}[y_j = 0] + \beta\, \text{peak_norm}_j$, where $\text{peak_norm}_j = \sqrt{y_j} / \max_{j'} \sqrt{y_{j'}}$ is the square-root magnitude of the observation at position j, normalized by the largest value in the history window. The zero penalty reduces, without removing, the influence of zero-valued keys, and the peak term raises the weight of rare high-magnitude observations. The summary state e_t is the layer-normalized representation at the last position of E_{enc}, and E_{enc} also serves as the key/value source for cross-attention in the decoders.

3.4 TARGET AND AUXILIARY DECODERS

Both decoders share a common structure: horizon-specific queries, built from a learned horizon embedding and the projected future-known context, pass through a stack of dilated convolutions

along the horizon axis and then query E_{enc} via cross-attention. These convolutions need not be causal, since the future-known context is available for the whole forecast window. The two decoders use the same architecture but do not share parameters.

Target decoder. Each horizon's query attends over E_{enc} through cross-attention, with a residual connection and layer normalization, producing the horizon-specific representation that is combined with z (Section 3.5).

Auxiliary decoder (optional). When a causally relevant process A (e.g., availability or exposure) is observed historically but not realized at forecast time, the auxiliary decoder predicts its future trajectory $\hat{A}_{i,t+1:t+H}$ from E_{enc} and the future-known context. It is trained jointly with the main objective on the observed process. The target decoder receives this prediction instead of the realized future value, which is unobserved at forecast time; Appendix C explains why this avoids look-ahead leakage. The prediction also indicates which horizons are expected to be active: when the auxiliary decoder is enabled, the cross-attention score of the target decoder gains a bias, $s_{h,j} = q_h^\top k_j/\sqrt{d} + \rho\,\omega_h\,\text{peak_norm}_j$, where peak_norm_j is defined in Section 3.3 and ω_h is a horizon-dependent activity score computed from $\hat{A}_{i,t+h}$, so that historical peaks receive more weight in the horizons where the predicted activity is relatively high. When no such process is observed, the auxiliary decoder and this bias are omitted.

3.5 Context-Conditioned Latent Regime

We inject the latent regime z through an epinet (Osband et al., 2023). An epinet reads a stop-gradient (sg) representation together with z; it consists of a trainable network and a fixed prior network with no trainable parameters, and its output is the inner product of their combined outputs with z. Ours has two branches, one reading the encoder summary e_t and one reading the target decoder's representation r_h of each horizon h, so that the effect of z at horizon h can depend on both the history and the horizon-specific context. Both branches read the same z. In short, z shifts the parameters of the Negative Binomial at every horizon around a deterministic prediction: with $a_h \in \mathbb{R}^2$ the pre-activations of its mean and dispersion,

$$a_h = b_h + \delta_h^{\text{enc}}\big(\text{sg}(e_t), z\big) + \delta_h^{\text{dec}}\big(\text{sg}(r_h), z\big), \qquad \delta(x,z) = \big(\psi_\eta^L(x,z) + \kappa\,\psi^P(x,z)\big)^\top z,$$

where b_h is the deterministic prediction, δ is the shift induced by z, ψ_η^L is the trainable network, ψ^P is the fixed prior network, and κ is a fixed scale; The mean and dispersion are then $\mu_h = \text{softplus}(a_h^{(\mu)})$ and $\alpha_h = \text{softplus}(a_h^{(\alpha)}) + \varepsilon$, the parameters of the Negative Binomial in Section 3.6.

The regime follows a context-conditioned Gaussian, $z_{i,t} \sim \mathcal{N}(m_z, \text{diag}(s_z)^2)$ with $(m_z, s_z) = q_\phi\big(\text{sg}(e_t), \tilde{\mathcal{F}}_{i,t}\big)$, where $\tilde{\mathcal{F}}_{i,t}$ is the future-known context over the forecast window, extended with the relational signal $g_{i,t}$ and the auxiliary prediction $\hat{A}$ when these components are enabled. We use a Gaussian so that sampling can be reparameterized and trained by gradient descent, as in the original epinet. A single draw of z is shared across all H horizons and both branches, and each horizon reads the full vector z through its own coefficients.

Our use departs from the original epinet in two respects. First, the original draws its index from a fixed, input-independent reference distribution, whereas z is conditioned on the current forecast instance, since our goal is dependence across the horizons of a single series rather than joint predictions across independent inputs. Second, the epinet has two branches that read the same z. Because z is drawn from q_ϕ given the forecast context in both training and inference, there is no separate network that infers z from the target, and hence no KL divergence to align the two. Its parameters are trained with the prediction loss, together with a small regularizer on (m_z, s_z); details are given in Appendix D.

3.6 Joint Predictive Distribution

At each horizon, the parameters $\mu_{i,t+h}$ and $\alpha_{i,t+h}$ obtained in Section 3.5 define a Negative Binomial with mean $\mu_{i,t+h}$ and variance $\mu_{i,t+h} + \alpha_{i,t+h}\mu_{i,t+h}^2$:

$$Y_{i,t+h} \mid z_{i,t} \sim \text{NB}(\mu_{i,t+h}, \alpha_{i,t+h}).$$

The horizons are conditionally independent given $z_{i,t}$, but $z_{i,t}$ is shared across all H, so marginalizing over $z_{i,t}$ yields a joint predictive distribution under which the horizons are in general dependent,

$$p_\theta(Y_{i,t+1:t+H} \mid I_{i,t}) = \int \prod_{h=1}^{H} p_\theta(Y_{i,t+h} \mid z_{i,t}, I_{i,t})\, q_\phi(z_{i,t} \mid I_{i,t})\, dz_{i,t}.$$

This is the joint prediction of an epistemic neural network (Osband et al., 2023, Eq. 2), with the context-conditioned q_ϕ in place of a fixed reference distribution; Section 4.2 analyzes the dependence it induces.

We approximate the integral by Monte Carlo sampling of $z_{i,t}$. In training, we draw K samples $z_{i,t}^{(k)} \sim q_\phi$ and minimize the joint negative log-likelihood of the observed trajectory,

$$\mathcal{L}_{\text{joint}} = -\frac{1}{H} \log \frac{1}{K} \sum_{k=1}^{K} \prod_{h=1}^{H} \mathrm{NB}\big(y_{i,t+h}; \mu_{i,t+h}^{(k)}, \alpha_{i,t+h}^{(k)}\big),$$

which is combined with a tail-weighted marginal term and a small regularizer on (m_z, s_z) in the training objective given in Appendix D. At inference, each draw of $z_{i,t}$ is followed by an independent draw of $Y_{i,t+h}$ at every horizon. Horizon-wise quantiles are computed from the samples of each horizon, and trajectory-level quantities, such as the probability that all H horizons are zero, from the sampled trajectories.

3.7 Optional Relational Context

When the panel of N series (Section 3.1) contains related series, e.g., in the same category or with similar attributes, the framework optionally adds a relational signal $g_{i,t}$ that summarizes the recent behavior of a neighbor set $\mathcal{N}(i)$, such as the neighbors' average demand and their rates of active and zero periods. $g_{i,t}$ is computed using only information available up to the forecast origin t and is refreshed as t advances. It is appended to the future-known context (Section 3.2), so that the latent regime and the target decoder can draw on related series when a series' own history is too sparse to be informative. The networks that read $g_{i,t}$ can learn positive or negative dependence on it, so neighbors need not be assumed to move together with the series. This component is not required by the core architecture (Sections 3.3 to 3.6), which operates without it when supporting metadata is unavailable.

4 Theoretical Analysis

4.1 Zero as a Learnable Signal

A zero is just a number for "none," but where and how it appears in time can hint at what might be going on, even though the true cause is rarely known.

Observation 1. *A predictor that summarizes zero information only through $\hat{\pi}^{(0)}$ produces the same output for any two histories with the same zero rate, regardless of how their zeros are arranged in time.*

Example. Consider two ten-week histories, $\mathcal{H}_A = (1,1,1,1,1,0,0,0,0,0)$ and $\mathcal{H}_B = (0,1,0,1,0,1,1,0,0,1)$, where 1 marks a week with positive demand; both have the same zero rate, $1/2$. In $\mathcal{H}_A$, the zeros are bunched together at the end, as if the product went out of stock partway through and stayed that way. In $\mathcal{H}_B$, the zeros are scattered among positive weeks, as if the product stayed in stock and simply was not bought every week. These situations point to different futures: $\mathcal{H}_A$ looks likely to stay at zero, while $\mathcal{H}_B$ looks likely to keep alternating. A predictor that only tracks the zero rate would treat them identically, by Observation 1.

4.2 Expressiveness Gap Between Independent and Shared Regimes

Accurate horizon-wise marginal forecasts are not sufficient to identify the joint predictive distribution. Objectives that sum losses across horizons constrain the marginal distribution at each horizon, but do not determine how zero and positive events are jointly organized across the forecast window. Figure 1 gives a concrete example: shared dormant and active regimes can produce the same per-horizon marginals as independently varying regimes, while assigning very different probabilities to how zero and positive outcomes combine across the window.

Lemma 1 (Non-identifiability under marginal objectives)**.** *Suppose the forecasting objective decomposes across horizons, $\mathcal{L}_{\text{marg}} = \sum_{h=1}^{H} \mathbb{E}[\ell_h(Y_{t+h}, P_h)]$, where P_h is the predictive marginal distribution at horizon h. Any two joint predictive distributions with identical horizon-wise marginals attain the same value of $\mathcal{L}_{\text{marg}}$, regardless of their cross-horizon dependence structures.*

Proof sketch. Each term in $\mathcal{L}_{\text{marg}}$ depends only on the corresponding marginal P_h, so replacing the joint distribution while leaving all P_h unchanged cannot alter the objective. For instance, encoding a positive-demand period by 1: Model A assigns probability $1/2$ to each of $(0,0)$ and $(1,1)$, while Model B assigns probability $1/2$ to each of $(0,1)$ and $(1,0)$; both have $P(Y_{t+1} = 0) = P(Y_{t+2} = 0) = 1/2$, yet Model A assigns probability $1/2$ to two consecutive zeros and Model B assigns probability zero. □

This does not imply that the resulting marginal forecasts are inaccurate; rather, accurate marginals alone cannot determine whether future sparse events are persistent, alternating, or clustered. For two horizons, let $p_h = P(Y_{t+h} = 0 \mid I_t)$ and $p_k = P(Y_{t+k} = 0 \mid I_t)$. The marginals constrain the joint zero probability $q = P(Y_{t+h} = 0, Y_{t+k} = 0 \mid I_t)$ only through the Fréchet–Hoeffding bounds, $\max(p_h + p_k - 1, 0) \le q \le \min(p_h, p_k)$, and every value in this interval is attained by some joint distribution (Fréchet, 1951).

One natural way to represent this dependence is through a latent regime.

Definition 1 (Independent-regime model)**.** *A model is an* independent-regime model *if there exist latent variables $Z_1, \ldots, Z_H$, mutually independent given I_t, such that $Y_{t+1}, \ldots, Y_{t+H}$ are conditionally independent given $(Z_1, \ldots, Z_H, I_t)$ and the conditional law of Y_{t+h} depends on the latent variables only through Z_h.*

Lemma 2 (Dependence collapse under independent regimes)**.** *Consider the independent-regime model of Definition 1, in which $Z_1, \ldots, Z_H$ are conditionally independent given I_t, and the outcomes are conditionally independent given $Z_{1:H}$ and I_t. For any $h \neq k$ and any measurable functions g_h, g_k such that $g_h(Y_{t+h})$ and $g_k(Y_{t+k})$ have finite second moments, $\mathrm{Cov}(g_h(Y_{t+h}), g_k(Y_{t+k}) \mid I_t) = 0$.*

Proof sketch. Define $m_h(Z_h) = \mathbb{E}[g_h(Y_{t+h}) \mid Z_h, I_t]$. Applying the law of total covariance conditional on I_t, the covariance equals $\mathbb{E}[\mathrm{Cov}(g_h(Y_{t+h}), g_k(Y_{t+k}) \mid Z_{1:H}, I_t) \mid I_t] + \mathrm{Cov}(m_h(Z_h), m_k(Z_k) \mid I_t)$. The first term is zero by the conditional independence of the outcomes. The second is zero because $Z_h \perp Z_k \mid I_t$, and hence $m_h(Z_h)$ and $m_k(Z_k)$ are conditionally independent. A complete derivation is provided in Appendix B. □

The lemma applies both to demand values, by taking $g_h(y) = y$, and to zero events, by taking $g_h(y) = \mathbf{1}\{y = 0\}$. Thus, conditional on the observed information, an independent-regime model cannot represent stochastic dependence between zero events at different forecast horizons unless an additional shared source of randomness or an autoregressive dependence mechanism is introduced.

Lemma 3 (Dependence through a shared latent regime)**.** *Suppose that Y_{t+h} and Y_{t+k} are conditionally independent given a shared latent regime Z and I_t. For any measurable functions g_h, g_k such that $g_h(Y_{t+h})$ and $g_k(Y_{t+k})$ have finite second moments, define $m_h(Z) = \mathbb{E}[g_h(Y_{t+h}) \mid Z, I_t]$. Then, for any $h \neq k$, $\mathrm{Cov}(g_h(Y_{t+h}), g_k(Y_{t+k}) \mid I_t) = \mathrm{Cov}_Z(m_h(Z), m_k(Z) \mid I_t)$.*

Proof sketch. By the law of total covariance, the covariance equals $\mathbb{E}_Z[\mathrm{Cov}(g_h(Y_{t+h}), g_k(Y_{t+k}) \mid Z, I_t) \mid I_t] + \mathrm{Cov}_Z(m_h(Z), m_k(Z) \mid I_t)$. Conditional independence given Z makes the first term zero. A complete derivation is provided in Appendix B. □

Because the same realization of Z enters every horizon, variation in Z can move the horizon-wise conditional moments together. For raw demand, the model of Section 3.5 has $m_h(z) = \mathbb{E}[Y_{t+h} \mid z, I_t] = \mu_h(z)$. For zero events, define $u_h(z) = P(Y_{t+h} = 0 \mid z, I_t)$. Lemma 3 then gives

$$\mathrm{Cov}\left(\mathbf{1}\{Y_{t+h} = 0\},\, \mathbf{1}\{Y_{t+k} = 0\} \mid I_t\right) = \mathrm{Cov}_Z\left(u_h(Z),\, u_k(Z) \mid I_t\right).$$

A shared regime can therefore induce positive, negative, or zero dependence, depending on how the horizon-wise conditional probabilities vary with Z; importantly, unlike an independent-regime model, it is not restricted to zero conditional dependence.

Proposition 4.1 (Expressiveness gap)**.** *There exists a joint predictive distribution representable by the shared-regime model of Section 3.5 that no independent-regime model can represent, regardless of its training data or deterministic function capacity.*

Table 1: Full baseline comparison on the proprietary dataset.

	Ours	Core	**SPADE-S**	**Chronos2**	SFF	**TiDE**	CBolt	DLin	WaveNet	TFT	DOTheta	Theta	PatchTST	**AutoETS**	IMAPA	ADIDA	DeepAR	Naive	SNaive	ARIMA	RecTab	Croston	Avg	NPTS
P50	**.371**	.377	**.387**	**.387**	.407	**.409**	.412	.412	.427	.427	.437	.438	.445	**.447**	.453	.460	.486	.487	.487	.490	.492	.494	.556	.699
P70	**.386**	.395	**.402**	**.399**	.417	**.421**	.430	.425	.440	.448	.495	.493	.473	**.493**	.484	.493	.522	.636	.636	.531	.566	.546	.598	.752
P90	**.259**	.261	**.265**	**.262**	.284	**.294**	.292	.301	.305	.296	.328	.327	.359	**.314**	.304	.307	.437	.426	.426	.334	.364	.318	.416	.496

Lower is better. Methods are sorted left to right by P50. "Ours" and "Core" are the full model and the core-only variant (without the auxiliary decoder or relational context), respectively.

Proof. Consider two forecast horizons and let $Z \sim \mathcal{N}(m_z, s_z^2)$, $s_z > 0$. Conditional on Z and I_t, let the two outcomes be independent Negative Binomial variables with a common conditional mean $\mu_1(Z) = \mu_2(Z) = \mathrm{softplus}(a + bZ)$, $b \neq 0$, and any fixed positive dispersion parameter; this belongs to the shared-regime model class. By Lemma 3, $\mathrm{Cov}(Y_{t+1}, Y_{t+2} \mid I_t) = \mathrm{Var}_Z[\mathrm{softplus}(a + bZ) \mid I_t] > 0$, since $s_z > 0$ and $b \neq 0$ make the conditional mean nonconstant. An independent-regime model can draw Z_1, Z_2 independently from the same Gaussian and use the same decoder at both horizons, matching the two marginals exactly; but by Lemma 2, its cross-horizon covariance must be zero, so it cannot represent this joint distribution. □

The same distinction extends to the probability of an entirely zero forecast window: the shared and independent constructions above generally assign different probabilities to this event, even though every horizon-wise marginal is identical. This gap motivates both the shared-regime architecture and the joint training and evaluation criteria introduced in the following sections.

5 Experiments

Experimental setup. We evaluate both real-world applicability and the underlying mechanism (Section 5.2).For applicability, we use four real, large-scale datasets (Table 8 in Appendix H.2): a proprietary e-commerce demand dataset and three public benchmarks, Car Parts (Godahewa et al., 2020), Online Retail II (Chen, 2012), and M5 (Makridakis et al., 2022). Experiments on the proprietary dataset are conducted entirely offline, rather than within a live production deployment, and its auxiliary decoder is not used on the public datasets due to differing available features. Baseline configurations are given in Appendix H.

We adopt a rolling-origin protocol mimicking deployment: at each origin, the model observes history up to that point, forecasts the subsequent horizon, and the origin then advances. We report WAPE at $q \in \{0.5, 0.7, 0.9\}$, following the WQL formulation of prior work (Wolff et al., 2025): $\mathrm{WAPE}_q = \sum_i [q(y_i - \hat{y}_i)_+ + (1-q)(\hat{y}_i - y_i)_+] / \sum_i y_i$, pooled across all series and origins before dividing, where $\hat{y}_i$ is the predicted q-quantile for observation y_i. As a sum of horizon-wise quantile losses, WAPE is a marginal metric in the sense of Section 4.2; joint events are examined separately in Section 5.2. A predictor that always outputs zero yields $\mathrm{WAPE}_q = q$ exactly; our method beats this baseline across all datasets and quantiles. We report improvements over the strongest baseline in absolute WAPE points, with basis points (bps, $1\,\mathrm{bps} = 10^{-4}$) in parentheses for business interpretability.

5.1 Empirical Results

Proprietary dataset. We evaluate on the proprietary dataset and report WAPE at $q \in \{0.5, 0.7, 0.9\}$ against all baselines tested, spanning the method families discussed in Section 2 (Table 1). Our method achieves the best WAPE at every quantile, outperforming both the strongest pretrained foundation model, Chronos2, and SPADE-S, a method specifically designed for sparse demand. General-purpose neural architectures such as TiDE, along with classical statistical methods such as AutoETS and traditional intermittent-demand heuristics, perform considerably worse.

Public benchmarks. We conduct the same evaluation on the three public benchmarks, and our method achieves the best WAPE at every quantile on all three (Table 2). As with the proprietary dataset, Chronos2 remains strong on Online Retail II and M5. Of the three public benchmarks, Car Parts is the only one with no exogenous features available, and it is also the sparsest (79.2% zeros). On this dataset, Chronos2's performance drops to the middle of the ranking, suggesting that even foundation models such as Chronos2 can struggle when historical signal is limited and covariates are unavailable. Classical statistical methods and traditional intermittent-demand heuristics remain consistently behind across all three datasets. Notably, at P50 on Car Parts, only our method and DeepAR outperform the all-zero forecast baseline, while every other method fails to do so; this may reflect the fact that, in an extremely sparse setting without covariates, most methods struggle

to move past reliance on their own conventional historical signal, whereas DeepAR's autoregressive sampling and our explicitly modeled, zero-dominated latent regime both capture the near-all-zero structure more directly, allowing them to keep pace with this simplest baseline. Full experimental settings and model configurations are provided in Appendix H.

Table 2: Full baseline comparison across the three public benchmarks, sorted by Car Parts P50.

	Car Parts			Online Retail II			M5		
Model	P50	P70	P90	P50	P70	P90	P50	P70	P90
Ours	**.488**	**.617**	**.517**	**.329**	**.364**	**.289**	**.335**	**.354**	**.223**
DeepAR	.499	.705	.553	.498	.670	.731	.355	.372	.250
Zero Forecast*	.500	.700	.900	.500	.700	.900	.500	.700	.900
TFT	.509	.655	.581	.469	.630	.785	.363	.379	.252
TiDE	.517	.663	.596	.355	.402	.350	.365	.376	.238
DLinear	.524	.683	.660	.350	.392	.341	.366	.370	.260
DeepVAR-Copula	.541	.762	.587	.384	.437	.376	.366	.432	.264
PatchTST	.542	.665	.556	.334	.371	.300	.360	.371	.235
Autoformer	.582	.677	.552	.380	.435	.365	.382	.411	.253
Chronos2	.584	.744	.610	.332	.370	.305	.341	.357	.234
SBA	.612	.717	.539	.419	.477	.376	.435	.408	.289
ChronosBolt	.613	.733	.597	.338	.378	.296	.347	.366	.242
TSB	.615	.665	.545	.372	.432	.358	.377	.381	.271
GPVAR-Copula	.680	.801	.553	.350	.395	.335	.413	.441	.268
AutoETS	.714	.802	.533	.362	.419	.324	.385	.391	.240
DynamicOptimizedTheta	.735	.785	.538	.354	.389	.303	.382	.388	.240
SeasonalNaive	.799	.771	.593	.470	.540	.399	.444	.456	.305

Lower is better. P50/P70/P90 denote WAPE_q at $q = 0.5, 0.7, 0.9$ respectively. *Zero Forecast is a theoretical reference (always predicting zero), not a trained model; it satisfies $\text{WAPE}_q = q$ exactly by construction.

5.2 Ablations

We conduct controlled ablations on synthetic data generated from a known two-regime Hurdle-Negative-Binomial process, which allows us to isolate the effect of each architectural and objective choice against a known ground truth rather than the confounds present in real data. Models are evaluated at horizons $H \in \{3, 6, 12\}$. All comparisons hold the training objective and data splits fixed except for the ablated factor, and are averaged over three random seeds. We define the joint event of persistent inactivity across the full H-step forecast window, $D_i = \mathbf{1}\{Y_{i,t+1} = \cdots = Y_{i,t+H} = 0\}$, and let $\hat{p}_i = P(D_i = 1 \mid I_{i,t})$ denote the model's predicted probability of this event, estimated as the fraction of sampled trajectories that are zero at every horizon (Section 3.6) and clipped to $[\epsilon, 1-\epsilon]$ for numerical stability. Alongside horizon-wise WAPE at P50/P70/P90, we report the Brier score and log score of $\hat{p}_i$ against the empirical occurrence of D_i over N_{eval} evaluated (series, origin) pairs: $\text{Brier} = N_{\text{eval}}^{-1} \sum_i (\hat{p}_i - D_i)^2$ and $\text{LogScore} = -N_{\text{eval}}^{-1} \sum_i [D_i \log \hat{p}_i + (1 - D_i) \log(1 - \hat{p}_i)]$.

Temporal encoder. We compare three temporal encoder backbones at $H = 3$, evaluating each encoder's ability to predict per-horizon zero-demand occurrence, $O_{t+h} = \mathbf{1}(Y_{t+h} = 0)$. The three configurations differ only in the encoder: a 4-layer dilated causal convolution, a 2-layer LSTM, and a 2-layer Transformer encoder with 4 attention heads; all other architecture and training details are held fixed. TCN obtains the highest zero recall, precision, and F1 of the three (Table 5 in Appendix E), consistent with our choice of dilated convolutions (Section 3.3), although the differences are small.

Shared, independent, and no latent regime. We compare the full model (Full-SharedZ), a per-horizon independent regime (Independent-Z, Definition 1), and no regime at all (No-Z), at $H = 3, 6, 12$; full numerical results and the corresponding figure are given in Appendix E. At $H = 3$, the differences between variants are small. As H grows, Full-SharedZ improves on both metrics, while Independent-Z and No-Z worsen on log score; at $H = 12$, their log score (0.7651 and 0.7774)

exceeds that of a trivial predictor assigning probability 0.5 to every event ($\ln 2 \approx 0.693$), suggesting these variants are not merely uninformative but systematically miscalibrated at this horizon.

Context-conditioned versus fixed latent regime. At $H = 12$, we compare the context-conditioned regime against a variant where $z \sim \mathcal{N}(0, I)$ is sampled independently of the forecast context I_t, as in the original epinet (Osband et al., 2023) (Section 3.5). All-zero Brier and log score are computed as defined above.

The context-conditioned regime is better than the fixed-distribution variant on every metric reported, including a 10.2% relative reduction in all-zero log score. This result suggests that conditioning z on the forecast context, rather than sampling it independently as in the original epinet (Osband et al., 2023), allows the model to adapt the regime to the specific instance being forecast, rather than merely inducing generic across-sample dependence.

Table 3: Context-conditioned vs. fixed regime

Regime	Brier ↓	Log Sc. ↓	Energy ↓
Context-Cond.	**.169**$\pm$.01	**.54**$\pm$.037	**2.33**$\pm$.02
Fixed-Gauss.	.182$\pm$.006	.60$\pm$.019	2.36$\pm$.022

Output distribution head. At $H = 3$, we compare Negative Binomial against Hurdle-NB, ZINB, and Tweedie heads, with the shared regime held fixed. NB achieves the best WAPE at all three quantiles (Table 6 in Appendix E) and is our default head. The choice of output distribution has some effect on WAPE, but the differences across heads are modest compared to those from the shared latent regime.

Training objective on real data. To test whether training with the joint likelihood improves calibration beyond what architectural sharing of z provides on its own, and whether this holds on real, not only synthetic, data, we compare Marginal-only training ($\gamma = 0$) against our default Marginal+Joint training on Car Parts at $H = 6$, with 3 seeds.

Marginal+Joint training improves both WAPE and both joint-event metrics over Marginal-only training, confirming that training with the joint likelihood provides value beyond architectural sharing alone, and that this holds on real data.

Table 4: Marginal-only vs. Marginal+Joint

Objective	P50	P70	P90	Brier	Log Sc.
Marginal-only	.502 $\pm$.001	.634 $\pm$.002	.541 $\pm$.001	.204 $\pm$.003	.598 $\pm$.009
Marg.+Joint	**.487** $\pm$.002	**.619** $\pm$.002	**.530** $\pm$.002	**.189** $\pm$.000	**.556** $\pm$.001

Beyond these results, several points are worth discussing further, including the practical significance of the reported improvements, the role of the latent-moment regularizer, and directions for future work on the interpretability of the learned regime. We discuss each in Appendix G.

6 Conclusion

We proposed a regime-aware framework for joint probabilistic forecasting, built on the view that, unlike prior work, zeros carry information about an underlying state that should be modeled as a continuous latent regime shared across the forecast horizon. We showed theoretically that marginal objectives cannot identify how sparse events are jointly organized across a forecast window, that a shared latent regime is sufficient to induce this dependence, and that the independent per-horizon regimes underlying common architectures cannot express it regardless of training. This work began with a real operational problem, traced it to its root cause, and returned to that same setting for validation: across the proprietary dataset and three public benchmarks spanning different granularities and feature regimes, we showed consistent improvements over more than ten baselines, and joint-level calibration metrics confirmed the direction of these theoretical predictions. In a setting where forecast accuracy translates directly into inventory cost and stockout loss, correctly capturing the joint structure of sparse demand is not merely a theoretical refinement but a practical necessity at scale.

REFERENCES

Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, et al. Gluonts: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. 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, et al. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Daqing Chen. Online Retail II. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5CG6D.

John D Croston. Forecasting and stock control for intermittent demands. *Journal of the operational research society*, 23(3):289–303, 1972.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023.

Robert Fildes, Shaohui Ma, and Stephan Kolassa. Retail forecasting: Research and practice. *International Journal of Forecasting*, 38(4):1283–1318, 2022.

Jose A Fiorucci, Tiago R Pellegrini, Francisco Louzada, Fotios Petropoulos, and Anne B Koehler. Models for optimising the theta method and their relationship to state space models. *International journal of forecasting*, 32(4):1151–1161, 2016.

Maurice Fréchet. Sur les tableaux de corrélation dont les marges sont données. *Ann. Univ. Lyon, 3ˆ e serie, Sciences, Sect. A*, 14:53–77, 1951.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Car parts dataset (without missing values), 2020. URL `https://doi.org/10.5281/zenodo.4656021`.

Rob J Hyndman and Yeasmin Khandakar. Automatic time series forecasting: the forecast package for r. *Journal of statistical software*, 27:1–22, 2008.

Rob J Hyndman, Anne B Koehler, Ralph D Snyder, and Simone Grose. A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of forecasting*, 18(3):439–454, 2002.

Diane Lambert. Zero-inflated poisson regression, with an application to defects in manufacturing. *Technometrics*, 34(1):1–14, 1992.

Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International journal of forecasting*, 37(4): 1748–1764, 2021.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International journal of forecasting*, 38(4):1346–1364, 2022.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Konstantinos Nikolopoulos, Aris A Syntetos, John E Boylan, Fotios Petropoulos, and Vassilis Assimakopoulos. An aggregate–disaggregate intermittent demand approach (adida) to forecasting: an empirical proposition and analysis. *Journal of the Operational Research Society*, 62(3):544–554, 2011.

Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy. Epistemic neural networks. In *Advances in Neural Information Processing Systems*, volume 36, pp. 2795–2823, 2023.

Fotios Petropoulos and Nikolaos Kourentzes. Forecast combinations for intermittent demand. *Journal of the operational research society*, 66(6):914–924, 2015.

David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. *Advances in neural information processing systems*, 32, 2019.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191, 2020.

Oleksandr Shchur, Ali Caner Turkmen, Nick Erickson, Huibin Shen, Alexander Shirkov, Tony Hu, and Bernie Wang. Autogluon–timeseries: Automl for probabilistic time series forecasting. In *International Conference on Automated Machine Learning*, pp. 9–1. PMLR, 2023.

Aris A Syntetos and John E Boylan. The accuracy of intermittent demand estimates. *International Journal of forecasting*, 21(2):303–314, 2005.

Ruud H Teunter, Aris A Syntetos, and M Zied Babai. Intermittent demand: Linking forecasting to inventory obsolescence. *European Journal of Operational Research*, 214(3):606–615, 2011.

Maurice CK Tweedie et al. An index which distinguishes between some important exponential families. In *Statistics: Applications and new directions: Proc. Indian statistical institute golden Jubilee International conference*, volume 579, pp. 579–604, 1984.

Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12(1), 2016.

Malcolm Wolff, Matthew Li, Ravi Kiran Selvam, Hanjing Zhu, Kin G Olivares, Ruijun Ma, Abhinav Katoch, Shankar Ramasubramanian, Mengfei Cao, Roberto Bandarra, et al. Spade-s: A sparsity-robust foundational forecaster. *arXiv preprint arXiv:2507.21155*, 2025.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

LLM USAGE STATEMENT

In this work, we used generative AI tools for writing-related tasks, including grammar checking, wording adjustments, layout and table formatting, and LaTeX formula formatting, as well as improving the readability and structure of the paper. All cited literature was read by the authors, not solely by LLMs. Generative AI tools assisted us in setting up baselines and tuning hyperparameters. All theoretical claims, proofs, and experimental designs were proposed by the authors. LLMs were not used to define the research problem, propose the core idea, or create the algorithm and model architecture. We take responsibility for the final content of this work.

A GENERAL-PURPOSE FORECASTING BASELINES

This appendix further explains the basic modeling logic behind each baseline family summarized in Section 2.

Classical statistical methods. AutoETS (Hyndman et al., 2002), DynamicOptimizedTheta (Fiorucci et al., 2016), and AutoARIMA (Hyndman & Khandakar, 2008) are based on exponential smoothing, the Theta method, and ARIMA modeling respectively, each automatically selecting model structure or parameters per series. NPTS (Alexandrov et al., 2020) is a non-parametric method that generates forecasts by weighted resampling of a single series' historical observations. RecursiveTabular (Shchur et al., 2023) instead converts historical targets into lag features, trains a shared regression model across the entire panel, and generates multi-step forecasts recursively.

Recurrent and attention-based neural models. DeepAR (Salinas et al., 2020) models history with a recurrent network and generates future trajectories through autoregressive sampling. TFT (Lim et al., 2021) combines recurrent layers, variable selection, gating mechanisms, and attention to handle static features, historical variables, and known future covariates. PatchTST (Nie et al., 2022) splits the series into contiguous time patches and applies self-attention over these patches. Autoformer (Wu et al., 2021) combines series decomposition with a periodicity-based Auto-Correlation mechanism to capture trend, seasonality, and long-range temporal dependence.

Convolutional and linear/MLP-based neural models. WaveNet (Van Den Oord et al., 2016) uses dilated causal convolutions to cover a long history while preserving causality. DLinear (Zeng et al., 2023) decomposes the series into trend and seasonal components and produces forecasts through separate linear mappings for each. SimpleFeedForward (Alexandrov et al., 2020) uses a fully connected network to map a fixed-length history window to a future predictive distribution. TiDE (Das et al., 2023) is likewise MLP-based, but further incorporates residual structure, covariate encoding, and a multi-step decoding component.

Pretrained time series foundation models. Chronos (Ansari et al., 2024), Chronos-Bolt, and Chronos2 (Ansari et al., 2025) all follow the paradigm of pretraining on a large-scale corpus of time series and then transferring to new forecasting tasks. The original Chronos quantizes numerical sequences into tokens and generates future trajectories autoregressively; Chronos-Bolt uses a patch-based representation and directly outputs multi-step quantile forecasts to improve inference efficiency; Chronos2 further supports univariate, multivariate, and covariate-conditioned forecasting tasks. We use these pretrained models zero-shot, with frozen weights, or with task-specific fine-tuning, following the specific experimental protocol for each.

B FULL DERIVATIONS

This appendix provides the complete derivations underlying the proof sketches of Lemmas 2 and 3 in Section 4.2.

Lemma 4 (Conditional law of total covariance)**.** *Let X and Y be random variables with finite second moments, and let Z and W be conditioning variables. Then*

$$\mathrm{Cov}(X, Y \mid W) = \mathbb{E}\big[\,\mathrm{Cov}(X, Y \mid Z, W) \mid W\big] + \mathrm{Cov}\left(\mathbb{E}[X \mid Z, W],\ \mathbb{E}[Y \mid Z, W] \mid W\right).$$

Proof. By the tower property,

$$\mathbb{E}\big[\,\mathrm{Cov}(X, Y \mid Z, W) \mid W\big] = \mathbb{E}[XY \mid W] - \mathbb{E}\big[\mathbb{E}[X \mid Z, W]\,\mathbb{E}[Y \mid Z, W] \mid W\big].$$

Moreover,

$$\mathrm{Cov}\left(\mathbb{E}[X \mid Z, W], \mathbb{E}[Y \mid Z, W] \mid W\right) = \mathbb{E}\big[\mathbb{E}[X \mid Z, W]\,\mathbb{E}[Y \mid Z, W] \mid W\big] - \mathbb{E}[X \mid W]\,\mathbb{E}[Y \mid W],$$

where the tower property gives $\mathbb{E}[\mathbb{E}[X \mid Z, W] \mid W] = \mathbb{E}[X \mid W]$, and analogously for Y. Adding the two expressions cancels the middle term and yields $\mathbb{E}[XY \mid W] - \mathbb{E}[X \mid W]\,\mathbb{E}[Y \mid W] = \mathrm{Cov}(X, Y \mid W)$. ■

B.1 Proof of Lemma 2

Fix $h \neq k$ and measurable functions g_h and g_k such that $g_h(Y_{t+h})$ and $g_k(Y_{t+k})$ have finite second moments. Define $m_h(Z_h) = \mathbb{E}[g_h(Y_{t+h}) \mid Z_h, I_t]$ and $m_k(Z_k) = \mathbb{E}[g_k(Y_{t+k}) \mid Z_k, I_t]$.

By Definition 1, the conditional joint distribution factorizes across horizons:

$$p(Y_{t+1:t+H}, Z_{1:H} \mid I_t) = \prod_{j=1}^{H} p(Z_j \mid I_t)\, p(Y_{t+j} \mid Z_j, I_t).$$

Consequently, $\mathbb{E}[g_h(Y_{t+h}) \mid Z_{1:H}, I_t] = m_h(Z_h)$, and analogously for horizon k.

Applying Lemma 4 with $X = g_h(Y_{t+h})$, $Y = g_k(Y_{t+k})$, $Z = Z_{1:H}$, $W = I_t$, gives

$$\mathrm{Cov}(g_h(Y_{t+h}), g_k(Y_{t+k}) \mid I_t) = \mathbb{E}\big[\,\mathrm{Cov}(g_h(Y_{t+h}), g_k(Y_{t+k}) \mid Z_{1:H}, I_t) \mid I_t\big] + \mathrm{Cov}(m_h(Z_h), m_k(Z_k) \mid I_t).$$

The first term is zero because the outcomes are conditionally independent given $Z_{1:H}$ and I_t. For the second term, Definition 1 gives $Z_h \perp Z_k \mid I_t$. Conditional independence is preserved under measurable transformations, so $m_h(Z_h) \perp m_k(Z_k) \mid I_t$. It follows that $\mathbb{E}[m_h(Z_h) m_k(Z_k) \mid I_t] = \mathbb{E}[m_h(Z_h) \mid I_t]\,\mathbb{E}[m_k(Z_k) \mid I_t]$, and hence $\mathrm{Cov}(m_h(Z_h), m_k(Z_k) \mid I_t) = 0$. Both terms therefore vanish, yielding $\mathrm{Cov}(g_h(Y_{t+h}), g_k(Y_{t+k}) \mid I_t) = 0$. ■

B.2 Proof of Lemma 3

Fix $h \neq k$ and measurable functions g_h and g_k such that $g_h(Y_{t+h})$ and $g_k(Y_{t+k})$ have finite second moments. Define $m_h(Z) = \mathbb{E}[g_h(Y_{t+h}) \mid Z, I_t]$ and $m_k(Z) = \mathbb{E}[g_k(Y_{t+k}) \mid Z, I_t]$.

Applying Lemma 4 with $X = g_h(Y_{t+h})$, $Y = g_k(Y_{t+k})$, $W = I_t$, and conditioning additionally on the shared latent variable Z, gives

$$\mathrm{Cov}(g_h(Y_{t+h}), g_k(Y_{t+k}) \mid I_t) = \mathbb{E}\big[\,\mathrm{Cov}(g_h(Y_{t+h}), g_k(Y_{t+k}) \mid Z, I_t) \mid I_t\big] + \mathrm{Cov}(m_h(Z), m_k(Z) \mid I_t).$$

By assumption, $Y_{t+h} \perp Y_{t+k} \mid Z, I_t$, so measurable transformations of these variables are also conditionally independent. Therefore, $\mathrm{Cov}(g_h(Y_{t+h}), g_k(Y_{t+k}) \mid Z, I_t) = 0$. The first term vanishes, leaving

$$\mathrm{Cov}(g_h(Y_{t+h}), g_k(Y_{t+k}) \mid I_t) = \mathrm{Cov}(m_h(Z), m_k(Z) \mid I_t). \qquad \blacksquare$$

C Forecast-Origin Admissibility

This appendix supports the use of the auxiliary prediction $\hat{A}$ (Section 3.4), which is used only on the proprietary data. At forecast time the realized trajectory $A_{t+1:t+H}$ is not observed, so a predictor trained with it would rely on information that is unavailable at inference.

Definition 2 (Forecast-origin admissibility)**.** *A predictor ψ is* forecast-origin admissible *if it is a measurable function of I_t alone, computable identically at training and inference.*

Let $\psi_{\mathrm{real}}(I_t, A_{t+1:t+H})$ use the realized trajectory, and $\psi_{\mathrm{pred}}(I_t) = \psi(I_t, \hat{A}_{t+1:t+H})$ use the predicted trajectory of Section 3.4, a function of I_t alone.

Fact. ψ_{pred} is admissible and ψ_{real} is not: $A_{t+1:t+H}$ is unobserved at inference and must be replaced by a proxy. Since $\sigma(I_t)$-measurable predictors are a subset of $\sigma(I_t, A_{t+h})$-measurable ones, the best achievable risk of the former is no better than that of the latter; if A_{t+h} carries information about Y_{t+h} beyond I_t, the gap is strict, so training with the realized A_{t+h} underestimates deployment risk. ψ_{pred} uses only $\sigma(I_t)$ throughout, so under the same data distribution its expected training risk equals its expected deployment risk.

D LATENT REGIME IMPLEMENTATION DETAILS

This appendix specifies the training objective, architecture, and sampling procedure for the latent regime $z_{i,t}$ (Section 3.5), to support reproducibility.

Posterior parameterization. The regime is drawn from $z_{i,t} \sim \mathcal{N}(m_z, \mathrm{diag}(s_z)^2)$ with latent dimension $d_z = 16$. The standard deviation is parameterized as $s_z = \mathrm{softplus}(\ell_z) + \epsilon$ for a network output ℓ_z and fixed floor $\epsilon = 10^{-4}$, which architecturally prevents s_z from collapsing to exactly zero, though it does not penalize s_z from approaching this floor (see the discussion below). Sampling uses the reparameterization trick, $z = m_z + s_z \odot \epsilon'$ with $\epsilon' \sim \mathcal{N}(0, I)$.

Training objective. The full training loss is a weighted combination of a marginal and a joint term, together with a latent regularizer:

$$\mathcal{L} = (1-\gamma)\mathcal{L}_{\mathrm{NB}} + \gamma\mathcal{L}_{\mathrm{joint}} + \lambda_{\mathrm{reg}}\mathcal{R}_z,$$

where $\mathcal{L}_{\mathrm{NB}}$ is the tail-weighted Negative Binomial negative log-likelihood applied to horizon-wise marginals, and $\mathcal{L}_{\mathrm{joint}}$ is a Monte Carlo estimate of the joint mixture log-likelihood over complete forecast trajectories,

$$\mathcal{L}_{\mathrm{joint}} = -\frac{1}{BH}\sum_{i=1}^{B}\log\left[\frac{1}{n_Z}\sum_{m=1}^{n_Z}\prod_{h=1}^{H} p_{\mathrm{NB}}\left(y_{i,h} \mid \mu_{i,h}^{(m)}, \alpha_{i,h}^{(m)}\right)\right],$$

with each m corresponding to one latent-regime draw $z_i^{(m)}$. This term directly estimates the integral in the joint distribution formulation of Section 3.6 via Monte Carlo sampling over z, rather than relying solely on the architectural sharing of z to induce dependence implicitly. For numerical stability, the inner sum is computed via the log-sum-exp identity,

$$\log\left[\frac{1}{n_Z}\sum_{m=1}^{n_Z}\exp(\ell_i^{(m)})\right] = \mathrm{logsumexp}_m(\ell_i^{(m)}) - \log n_Z,$$

where $\ell_i^{(m)} = \sum_{h=1}^{H}\log p_{\mathrm{NB}}(y_{i,h} \mid \mu_{i,h}^{(m)}, \alpha_{i,h}^{(m)})$. The regularizer is $\mathcal{R}_z = \mathbb{E}[m_z^2 + s_z^2]$. No KL divergence to an explicit prior is used (Section 3.5), and quantile losses are not part of training: P50/P70/P90 are obtained only at inference by summarizing Monte Carlo samples. During training, $n_Z = 8$ draws of z are used per batch to estimate $\mathcal{L}_{\mathrm{joint}}$.

Epinet architecture. Both the encoder-side and decoder-side epinet modules (Section 3.5) consist of a trainable network and a frozen prior network of matching structure, combined via an elementwise product with z and summed. The encoder-side trainable network is a 2-layer MLP (hidden width 64, ReLU, dropout $p = 0.3$) operating on a stop-gradiented encoder summary state; its frozen prior network has the same structure without dropout, with a combination scale of 0.3. The decoder-side epinet operates analogously on a stop-gradiented per-horizon decoder representation, with a combination scale of 0.20. Dropout is active only during training; at inference the model is set to evaluation mode, disabling dropout so that predictive variability across Monte Carlo samples arises solely from the sampled regime z, not from stochastic dropout masks.

Sampling. At inference, $M = 100$ samples of z are drawn to estimate the joint predictive distribution and derive horizon-wise quantiles.

E ADDITIONAL ABLATION RESULTS

This appendix provides the full numerical results and figures underlying the ablations summarized in Section 5.2.

E.1 Temporal Encoder

Table 5: Zero-occurrence prediction by encoder backbone ($H = 3$).

Encoder	Recall	Precision	F1
TCN	**0.9711**	**0.8377**	**0.8995**
LSTM	0.9696	0.8080	0.8815
Transformer	0.9640	0.8108	0.8810

E.2 Shared, Independent, and No Latent Regime

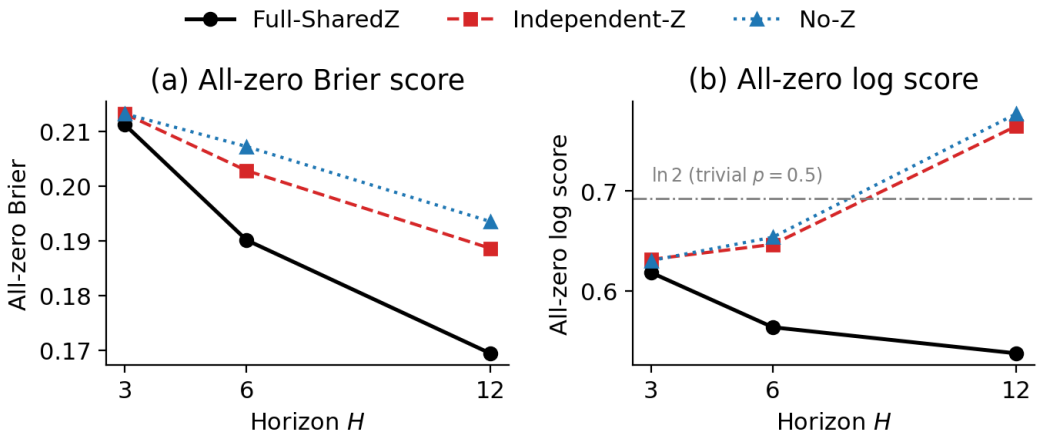

Figure 3: All-zero Brier score (a) and log score (b) versus forecast horizon H. The dash-dotted line in (b) marks the log score of a trivial predictor assigning probability 0.5 to every event ($\ln 2 \approx 0.693$).

E.3 Output Distribution Head

Table 6: WAPE by output distribution head ($H = 3$).

Distribution Head	P50	P70	P90
NB	**0.4716**±0.0024	**0.5758**±0.0020	**0.4509**±0.0037
Hurdle-NB	0.4766±0.0045	0.5815±0.0038	0.4589±0.0021
ZINB	0.4776±0.0032	0.5828±0.0038	0.4586±0.0045
Tweedie	0.4759±0.0019	0.5808±0.0017	0.4552±0.0034

F Computational Efficiency

We measure computational efficiency on the final rolling origin of Car Parts (2,674 series, 12-month history, 3-month horizon) using CPU execution. Each model is trained from scratch for 200 optimization steps, and we report training time per step. Model size is measured as total parameter count. Inference latency is measured over five runs after one warm-up run, using the median, and normalized to 1,000 series. Our model and DeepAR use $M = 100$ Monte Carlo samples to produce quantile forecasts; other baselines use their native quantile output heads directly.

Method	Parameters	200-step time	Time/step	Latency/1,000 series
DLinear	234	8.017 s	0.0401 s	0.0991 s
TiDE	643,169	63.460 s	0.3173 s	**0.0830 s**
PatchTST	71,113	122.135 s	0.6107 s	0.1800 s
TFT	56,336	60.870 s	0.3044 s	0.1899 s
DeepAR	4,546	11.839 s	0.0592 s	0.2464 s
Autoformer	80,259	357.341 s	1.7867 s	0.3867 s
Ours	**74,408**	**57.318 s**	**0.2865 s**	1.0958 s

Our model's training cost (0.2865s/step, 74,408 parameters) is comparable to or lower than several baselines (e.g., PatchTST, Autoformer). Inference latency, however, is the highest among compared methods (1.10s per 1,000 series), roughly $4\times$ that of DeepAR, the only other sampling-based method in this comparison ($M = 100$). This reflects both the shared cost of Monte Carlo sampling for joint distribution estimation and the additional computation from the dual epinet modules and encoder-decoder architecture. In latency-sensitive deployments, this cost can be reduced by lowering M at some expense to distributional accuracy, a trade-off we leave for future work to characterize systematically.

G Discussion

From joint dependence to marginal accuracy. The central theoretical claims of this paper concern joint dependence, yet the headline results in Section 5.1 are reported in WAPE, a marginal metric; this is the intended relationship, not a mismatch. The contribution is not that joint calibration improves in isolation, but that training with the joint likelihood improves the marginal predictions that deployment actually depends on. Three pieces connect this claim. The deployment objective is unambiguous: inventory and replenishment decisions depend on marginal quantile forecasts and WAPE, not on joint-event probabilities directly. The mechanism is motivated by theory: Sections 4.1–4.2 establish why zeros carry cross-horizon information that a marginal objective alone cannot exploit. And the effect is supported by real-data evidence: Table 4 shows that adding the joint term to an otherwise fixed architecture lowers WAPE at P50 from 0.502 to 0.487, with P70 and P90 improving as well, indicating the joint objective contributes value beyond architectural sharing alone. This is distinct from a claim of conceptual novelty: that joint training can improve marginal forecasts is not, by itself, a strong novelty claim. The contribution instead lies in how this is instantiated for sparse, zero-dominated series, and in demonstrating that the resulting gain is consistent and reliable on a real deployment-relevant task rather than only on synthetic constructions.

Interpreting WAPE improvements. The WAPE improvements reported in Section 5.1 may appear modest in relative terms, particularly given that WAPE's denominator is dominated by the aggregate demand volume across a large and heavily zero-inflated population. Their practical significance, however, depends on the unit economics of the items being forecast rather than on the percentage figure alone: the same relative reduction in error corresponds to very different absolute costs depending on the price and volume of the items involved. In forecasting systems operating at the scale typical of large-scale e-commerce and retail demand planning, such relative improvements would compound across a large population of predictions into a meaningful reduction in excess inventory and stockout-driven lost sales. Correctly interpreting such improvements also depends on the metric used to measure them: point-forecast accuracy metrics such as RMSE or MAE, still widely used to evaluate demand forecasts, can be misleading under extreme sparsity, since a forecaster that always predicts zero already achieves a low error on such metrics (Fildes et al., 2022).

Regularizing the latent regime. The shared latent variable gives the model a mechanism for expressing cross-horizon dependence, but larger latent variance does not necessarily lead to better forecasts. Given the operational importance of marginal quantile forecast accuracy, we aim to retain the flexibility of joint modeling while placing a mild constraint on the magnitude of the latent distribution. We therefore include a weak latent-moment regularizer (Appendix D) that penalizes excessively large context-dependent means and standard deviations, intended to improve optimization stability. This term does not fix the latent distribution, nor does it enforce nonzero randomness; the resulting latent distribution is jointly determined by the predictive loss and the regularization

penalty. We regard it as a scale-control measure, not as a mechanism that prevents latent collapse or guarantees improved WAPE.

Future work. A promising direction is improving the interpretability of the learned regime. Our motivating examples (Section 4.1) build intuition around human-interpretable states such as stock-out versus active demand; a regime that aligns with such states would offer more than improved calibration, it could be surfaced to demand planners as a signal distinguishing a stockout from a genuine demand slowdown, directly informing replenishment decisions rather than only improving forecast accuracy internally. Pursuing this interpretability is a natural next step connecting this work back to the practical setting that motivated it.

H Model and Training Hyperparameters

H.1 Proprietary Dataset

All AutoGluon-based baselines on the proprietary dataset were run with `AutoGluon-TimeSeries` (v1.5.0) under a single shared protocol; unless noted otherwise, each model uses the library's default hyperparameters. Two baselines fall outside this AutoGluon protocol: `SPADE-S` is the incumbent production forecasting system, evaluated as deployed rather than retrained here, and `Chronos2` is fine-tuned separately (see below).

Shared configuration. Forecast horizon `prediction_length`$= 3$ weeks (LT3); the target is weekly unit demand; probabilistic outputs are produced at nine quantile levels $\{0.1, 0.2, \ldots, 0.9\}$; the tuning/validation metric is weighted quantile loss (`WQL`). At each of the 23 rolling origins the models are refit on a fixed 130-week trailing context window (about two seasonal cycles), using `num_val_windows`$= 3$ internal validation windows (`AutoARIMA` uses 1). Out-of-stock weeks are treated as missing (target set to `NaN`) during training. Series with more than 50% out-of-stock weeks post-launch are excluded, and each retained series must have at least 6 non-out-of-stock weeks. Runs use a fixed random seed (2025). Calendar, product-category, and auxiliary/relational covariates are supplied to the models that accept exogenous inputs; the remaining models use the target history alone. After model selection, the chosen models are refit on the combined training and validation data before forecasting. Table 7 summarizes the per-model settings.

Table 7: Baseline configuration on the proprietary dataset. "Default" means AutoGluon-TimeSeries default hyperparameters under the shared protocol above.

Model	Settings
Naive, SeasonalNaive, Average	Default
AutoETS, Theta, DynamicOptimizedTheta, NPTS	Default
CrostonSBA, ADIDA, IMAPA	Default
RecursiveTabular	Default
AutoARIMA	Default
SimpleFeedForward, TFT, DeepAR, PatchTST, DLinear, TiDE, WaveNet	Default, with `target_scaler="mean_abs"`
Chronos-Bolt (`bolt_base`)	Pretrained, zero-shot (no fine-tuning)
Chronos2	Full fine-tuning (see text)

Most baselines use up to an 8-hour per-origin training budget (`time_limit`$= 28{,}800$ s). The deep-learning models additionally set `target_scaler="mean_abs"` (mean-absolute scaling), well suited to the heavy-tailed, high-zero-rate demand here; all other settings are AutoGluon defaults. `AutoARIMA` is run in a separate pass (per-series ARIMA fitting is slow) parallelized across all CPU cores (`n_jobs`$= -1$). `CrostonSBA` is the Syntetos–Boylan Approximation, i.e. the bias-corrected variant of Croston's method (not the original Croston estimator). `Chronos-Bolt` is applied zero-shot from the pretrained `bolt_base` checkpoint.

`SPADE-S` is a business forecasting system (a routing-based sparse forecaster,Wolff et al., 2025); we evaluate its deployed forecasts on the same rolling origins and series, without retraining or re-tuning it here.

Chronos2 is fine-tuned on the proprietary corpus rather than used zero-shot, using full fine-tuning (fine_tune_mode="full", in contrast to the library default LoRA adaptation), a fine-tuning learning rate of 1×10^{-6} (default 1×10^{-5}), 2,000 fine-tuning steps (default 1,000), robust target scaling (target_scaler="robust"), and held-out validation with early stopping during fine-tuning (eval_during_fine_tune=True), under a 4-hour training budget (time_limit= 14,400 s).

H.2 Public Benchmark Datasets

We evaluate the public-data baselines on Car Parts, Online Retail II, and M5 using a common leakage-free rolling-origin protocol. At each origin, only observations strictly preceding the forecast origin are available for training. Hyperparameters and stopping epochs are selected using a validation period that precedes the test origins and are then frozen. Trainable models are refit from scratch at every test origin. We report pooled global quantile-WAPE at quantiles $\{0.5, 0.7, 0.9\}$, aggregating penalties and realized demand over all series, origins, and forecast horizons before taking their ratio. Sampling-based models use 500 predictive trajectories for the main accuracy evaluation.

Table 8 summarizes the dataset-specific forecasting protocols.

Table 8: Dataset-specific configurations for the public benchmarks.

Dataset	Frequency	Context length	Horizon	Test origins
Car Parts	Monthly	12 months	3 months	10
Online Retail II	Weekly	26 weeks	3 weeks	10
M5	Daily	364 days	28 days	5

For Car Parts, we use the official no-missing release containing 2,674 monthly series and 51 observations per series. The publisher-replaced zeros are retained without additional imputation. The models use demand history only, with a fixed validation origin at month 36 and rolling test origins 39–48. Seasonal Naive uses a lag of 12 months. The smoothing parameters for Croston–SBA and TSB are selected on the fixed validation block from $\{0.05, 0.10, 0.20, 0.30\}$; TSB tunes both its occurrence and size parameters. AutoETS uses the nonseasonal ZZN specification because only 12 historical observations are supplied at each origin.

Online Retail II is aggregated into a weekly UK product panel indexed by StockCode. Each model receives the latest 26 weeks of demand and predicts the following 3 weeks over 10 rolling origins. M5 uses all 30,490 bottom-level item–store series, a 364-day context, a 28-day horizon, and five non-overlapping origins at days 1,801, 1,829, 1,857, 1,885, and 1,913. Calendar, event, SNAP, and selling-price variables are supplied to M5 models that accept known covariates; history-only models receive only past demand.

Neural baselines. Unless otherwise noted, trainable neural baselines are optimized for 180 epochs per rolling-origin refit, with a fixed seed and nonnegative output support. DLinear, TiDE, PatchTST, TFT, and Autoformer use their official NeuralForecast implementations, with the dataset-specific context and horizon from Table 8. The shared implementation uses robust target scaling, a batch size of 256, and a window batch size of 2,048. For Car Parts, DLinear uses a moving-average window of 3; PatchTST uses two encoder layers, four attention heads, hidden size 64, patch length 4, and stride 2; TFT uses hidden size 64, four heads, and dropout 0.1; and Autoformer uses hidden size 64, two encoder layers, one decoder layer, four heads, and a moving-average window of 3.

DeepAR uses a Negative-Binomial likelihood and generates 500 trajectories per forecast. On Car Parts, it uses two LSTM layers, hidden size 64, and dropout 0.1. DeepVAR and GPVAR use two recurrent layers with 40 cells per layer, dropout 0.1, learning rate 10^{-3}. Their low-rank covariance dimension is 10; GPVAR samples 20 target dimensions during training. Both models generate 500 coherent forecast trajectories. Chronos-Bolt-small is evaluated zero-shot from the pretrained Amazon checkpoint, without dataset-specific fine-tuning. For all models, unspecified settings follow the defaults of the corresponding official implementation.