Impact-Driven Event Embeddings for Context-Aware Forecasting and Anomaly Detection in Financial Time Series

Elham Alipour
ealipour@amazon.com
Amazon
Vancouver, British Columbia, Canada

Xiaoli Zhang
zhasabri@amazon.com
Amazon
Vancouver, British Columbia, Canada

Tom Boyang Jin
boyanjin@amazon.com
Amazon
Vancouver, British Columbia, Canada

Yu Gao
gaoiris@amazon.com
Amazon
Vancouver, British Columbia, Canada

Alex Moschos
amoschos@amazon.com
Amazon
Vancouver, British Columbia, Canada

Eugene Kim
eugeneki@amazon.com
Amazon
Vancouver, British Columbia, Canada

Miriam Teng
miriaten@amazon.com
Amazon
Vancouver, British Columbia, Canada

ABSTRACT

Financial transaction time series are strongly shaped by recurring external events such as holidays, promotional campaigns, and settlement cycles. In practice, however, anomaly detection systems often evaluate these series without explicitly modeling event context, leading to excessive false positives during predictable event-driven fluctuations. We propose a context-aware framework based on impact-driven event embeddings, where events are represented by the residual behavioral patterns they induce in transaction series rather than by labels alone. To study this setting under controlled conditions, we construct a synthetic benchmark calibrated to empirical signatures from production financial transaction data, spanning heterogeneous financial accounts, realistic event calendars, and injected anomalies. The framework estimates non-event baselines from observed series, learns compact event embeddings from event-centered residual responses, and uses them as future-known covariates in a probabilistic forecasting model. Anomalies are then defined as deviations from the resulting event-conditioned forecast distribution. In downstream experiments, the proposed approach outperforms both no-event and binary-event baselines for event-sensitive account profiles. Lag-bucketed embeddings are most effective for delayed-response cash accounts, while same-day embeddings are strongest for receivable and revenue accounts, where event effects are concentrated nearer the event itself. These results support the view that anomaly detection in financial time series is better framed as a context-aware forecasting problem in which learned event representations improve calibration and reduce false positives during known events.

KEYWORDS

financial time series, anomaly detection, forecasting, representation learning, event embeddings, synthetic data

1 INTRODUCTION

Financial transaction time series are shaped not only by regular operational rhythms, but also by recurring external events such as holidays, promotional campaigns, settlement cycles, payroll runs, and month-end close activities. These events can produce large but entirely expected shifts in transaction behavior. In practice, however, anomaly detection systems often evaluate such series without explicitly modeling event context. As a result, legitimate event-driven fluctuations are frequently flagged as anomalous, producing excessive false positives and reducing the usefulness of anomaly detection for financial monitoring and audit workflows. In large-scale financial environments, these false positives also create avoidable review burden and erode trust in anomaly alerts, especially when models must operate across heterogeneous and continuously changing account populations.

A central challenge is how to represent events in a way that is both informative and generalizable. Simple event indicators can mark that an event is active, but they provide little information about the type of response the event should induce, which account groups are likely to be affected, or when the effect should occur. Sparse categorical encodings are also limited: events with different names may induce similar financial behavior, while events with similar labels may affect different account families through very different mechanisms. In accounting data, this heterogeneity is especially important because cash, receivable, revenue, and accrual-oriented accounts often respond to the same event in distinct ways.

Learning event effects directly from each account's historical series is also often insufficient. Some accounts may have long histories with repeated event exposure, but many practical settings involve short transaction histories, sparse observations, or evolving account behavior. In such cases, an individual series may contain only one or a few occurrences of a given event, which makes account-specific event learning unreliable. This is particularly problematic in realistic financial environments where new accounts are continuously added and long training windows are not always available. A useful event representation should therefore support transfer across accounts, allowing models to benefit from shared event structure even when a specific series has limited historical coverage.

A second difficulty is that event effects are naturally defined relative to the event itself rather than by absolute calendar date. Some

important events occur on fixed dates, but many do not. Promotional events can shift across years, and religious or region-specific holidays may follow moving calendars. Even when the event identity is known in advance, the most informative dynamics often depend on event-relative timing: pre-event buildup, event-day disruption, delayed settlement, and post-event recovery. Models based only on absolute calendar features may capture broad seasonality, but they are less well suited to representing these event-centered response patterns.

Rigorous evaluation of event-aware anomaly detection is itself non-trivial: labeled financial transaction datasets are scarce, and existing benchmarks rarely capture the event-driven structure of real accounting systems. To study this setting under controlled conditions, we construct a synthetic benchmark spanning 1,000 accounts across four account families, with realistic event calendars and injected anomalies, calibrated to empirical signatures observed in production financial transaction data. This benchmark enables controlled study of both event-aware forecasting and downstream anomaly detection while preserving access to the underlying generative structure.

In this paper, we propose a framework for learning impact-driven event embeddings from residual response patterns induced by recurring events. The embeddings are constructed in event-relative time and used as future-known covariates in a probabilistic forecasting model, so anomalies are defined as deviations from an event-conditioned forecast distribution.

Our contributions are threefold. First, we introduce a synthetic benchmark for event-aware forecasting and anomaly detection in financial transaction series, calibrated to reflect key empirical signatures observed in production financial data and designed to capture heterogeneous account dynamics, realistic recurring events, lagged and asymmetric event effects, and richly labeled anomalies. Second, we develop a representation-learning pipeline that constructs event embeddings from event-centered residual response patterns, enabling models to share event knowledge across heterogeneous accounts rather than relying only on event labels or account-specific history. Third, we show that these learned event representations improve both forecasting and downstream anomaly detection relative to no-event and binary-event baselines, with different temporal alignments proving most effective for different account families.

Figure 1 summarizes the end-to-end framework, from synthetic and observed event-aware transaction data to residual-based event representation learning, event-conditioned forecasting, and downstream anomaly detection.

2 RELATED WORK

Our work connects financial anomaly detection, covariate-aware forecasting, pretrained forecasting models, and time-series representation learning.

2.1 Financial and accounting anomaly detection

Prior work on anomaly detection in financial and accounting data spans several methodological families. In accounting settings, Schreyer et al. [13] proposed adversarial autoencoder models for detecting anomalous accounting records, motivated by fraud detection and auditing use cases. Other financial time-series anomaly detection work has combined dimensionality reduction with neural scoring, for example through PCA-based representations followed by neural anomaly models [5]. Broader time-series anomaly detection literature has examined recurrent prediction and reconstruction-based methods such as LSTM-based detectors [11, 12], while audit and monitoring practice often relies on cluster-based, rule-based, or unsupervised approaches [8–10, 15].

These approaches demonstrate the value of compact representations and deviation-based scoring for complex financial behavior, but they generally treat anomalies as context-free deviations, record-level irregularities, or generic sequence-level outliers. They do not explicitly model known recurring external events or distinguish suspicious behavior from predictable event-driven fluctuations. In contrast, our work defines anomalies as deviations from an event-conditioned predictive distribution, grounding detection in the behavioral signatures induced by recurring events.

2.2 Forecasting with holidays, exogenous variables, and future-known covariates

Forecasting models with future-known covariates are closely related to our setting because event calendars are typically available in advance. Prophet [14] incorporates holiday effects through manually specified regressors and is therefore a natural practical benchmark. Recent deep forecasting models, including TimeXer [17], CITRAS [21], ETSformer [19], and TimesNet [20], also emphasize structured temporal variation and exogenous variables. However, these methods generally treat covariates as given inputs; they do not address how to construct transferable event features from realized financial response patterns. Our work differs in focusing not on a new generic forecasting architecture, but on learning event representations that can be supplied as future-known covariates in forecast-based anomaly detection.

2.3 Pretrained forecasting models

A particularly relevant recent line of work studies pretrained forecasting models that can be used with little or no task-specific retraining. Chronos [2] introduces pretrained probabilistic forecasting models based on tokenized time-series inputs, while Chronos-2 extends this paradigm to univariate, multivariate, and covariate-informed forecasting in a zero-shot setting [1]. Closely related, ChronosX [3] studies how covariates can be injected into pretrained time-series models through lightweight modular adapters. These models are directly relevant to our forecasting stage, which uses a Chronos-family backbone and future-known event covariates. Our contribution is complementary: rather than proposing a new generic forecaster, we study how event representations themselves can be learned from realized financial response patterns and then supplied to a probabilistic forecasting model.

2.4 Time-series representation learning

Our embedding stage is also related to contrastive representation learning for time series. TS2Vec [22] learns general time-series representations through hierarchical contrastive objectives, while CoST [18] uses contrastive learning to preserve disentangled temporal structure for forecasting. Unlike these works, we learn representations of event-response signatures rather than generic subseries

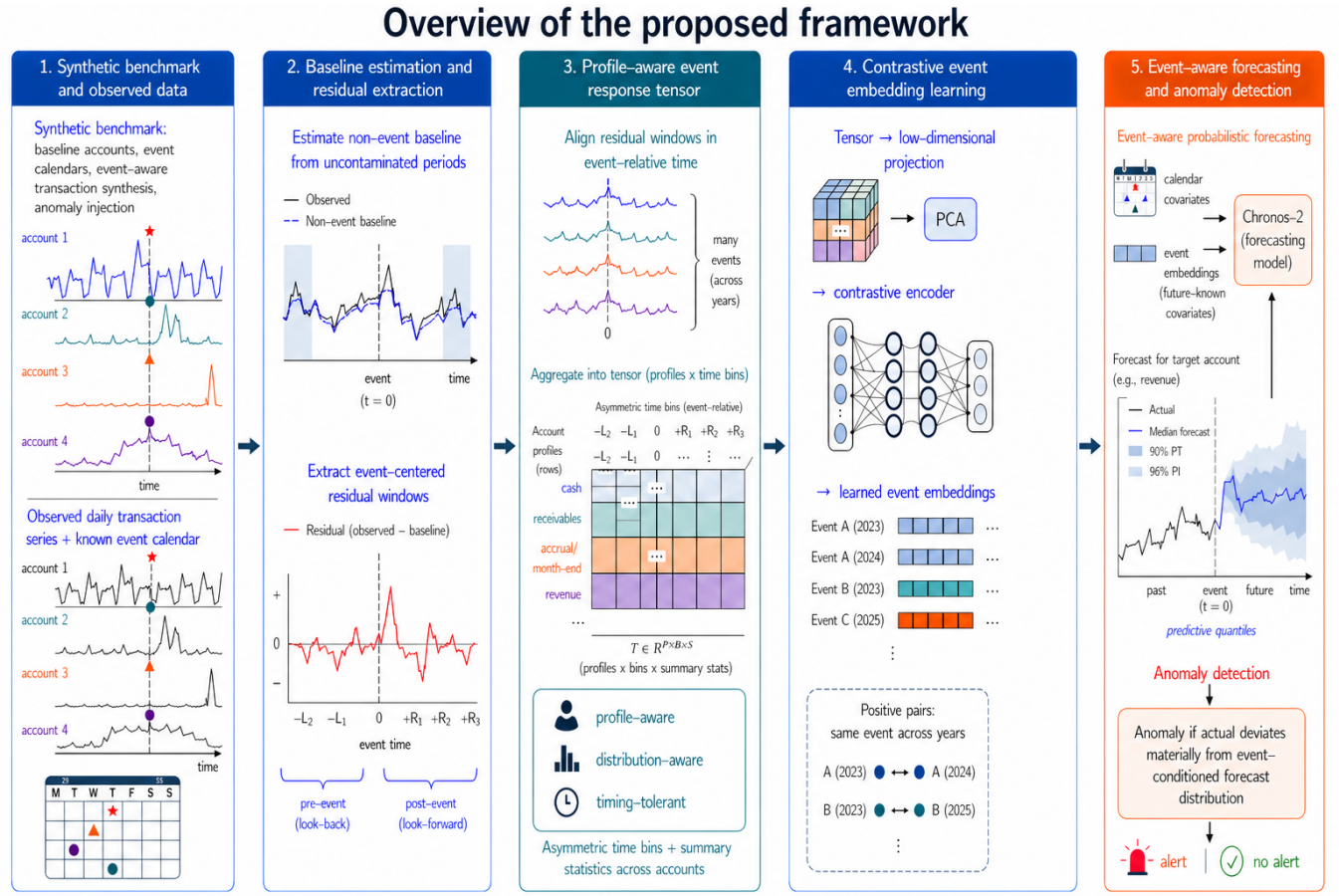

Figure 1: Overview of the proposed framework. The pipeline estimates non-event baselines, extracts event-centered residual responses, aggregates them into profile-aware event-response tensors, learns compact event embeddings through contrastive training, and uses the resulting event-conditioned forecasts for anomaly detection.

embeddings, and define positive pairs through cross-year recurrence of the same event rather than through generic augmentations.

3 SYNTHETIC BENCHMARK FOR EVENT-AWARE FINANCIAL TIME SERIES

3.1 Design goals

Public accounting transaction datasets with realistic event context and anomaly labels are scarce. In addition, in our application setting, company data-governance and confidentiality policies prevent release of the internal transaction data needed to construct a public benchmark directly from production records. To enable controlled evaluation despite these constraints, we construct a synthetic benchmark for event-aware forecasting and anomaly detection in financial time series. The benchmark is designed to reproduce key properties of real financial ledgers while preserving access to the latent generative factors, including baseline account behavior, event effects, and anomaly labels. This makes it possible to evaluate downstream models against known ground truth rather than incomplete proxy labels. Although the quantitative results reported in this paper are based on synthetic data, we also conducted parallel internal benchmarking on real-world data and observed the same qualitative pattern of improvement, giving additional confidence that the benchmark captures the event-driven structure relevant in practice.

In the experiments reported here, the benchmark spans three years of daily data (2023–2025) and 1,000 accounts in a U.S. retail and financial-operations setting. The current event layer includes major U.S. holidays and flagship retail events such as Black Friday, Cyber Monday and Back-to-School sale. The framework is not inherently U.S.-specific, however, and can be adapted to other regions by changing the event prompts, date rules, and validation constraints.

The benchmark's baseline account patterns and event-response mechanisms were calibrated against empirical analysis of production financial transaction data from the target application setting. In particular, we used real account-level summaries and qualitative response signatures across the four account families to anchor weekday structure, month-end concentration, activity sparsity, transaction-scale ranges, settlement lags, return timing, and event-driven response magnitudes. The resulting simulator remains

Table 1: Baseline account families in the synthetic benchmark.

Account family	Core temporal pattern	Event relevance
Cash / banking	Business-day concentration, low weekends, batching and settlement effects	High for holidays and promotions, often with delayed response
Receivables	Daily accumulation, settlement cycles, mild month-end structure	High for sales-driven and promotional events
Accrual / month-end	Sparse mid-month activity, sharp close-window spikes	Lower direct response; stronger downstream close effects
Revenue	High daily activity, weekly structure, seasonal peaks	High for holiday and promotional events

fully synthetic, but it is designed to reflect the statistical and temporal regimes observed in practice rather than an unconstrained hand-crafted world.

3.2 Baseline account generation

The baseline simulator models four account families: cash/banking, receivables, accrual/month-end, and revenue. These families capture distinct financial operating regimes. Cash accounts show strong business-day concentration and settlement-related patterns. Receivable accounts reflect sales accumulation and payment-settlement cycles. Accrual accounts concentrate activity near accounting close windows. Revenue accounts exhibit high daily volume, weekly structure, and strong sensitivity to seasonal and promotional demand.

Each account is first assigned a behavioral profile and then given account-specific parameters sampled from profile-dependent ranges. These parameters govern activity mode, weekly and monthly structure, sign policy, average transaction volume, count dispersion, amount scale, regularity, and long-term trend. Baseline transactions are then generated day by day using a staged process: an activity gate determines whether the account is active, the expected daily count is formed by combining baseline mean with weekly, monthly, and trend effects, realized counts are sampled from a negative binomial distribution, and transaction amounts are drawn from a lognormal distribution and signed according to account policy. This creates heterogeneous but interpretable account behavior before event effects are applied.

The profile definitions and parameter ranges were calibrated using empirical signatures extracted from production data, including weekday activity concentration, month-end intensity, transaction-scale variability, and sparsity patterns across the four account families.

3.3 Event calendar and impact-profile generation

The event layer is generated in two steps. First, an event calendar specifies which events occur in each year. Second, an account-type-specific impact profile specifies how each event affects transaction behavior. Known recurring events are date-anchored through calendar rules, while an LLM is used to generate structured yearly event lists and per-event impact profiles. For the U.S. benchmark used here, the prompts explicitly request major U.S. bank holidays, major retail events, and a small number of additional promotional events distributed through the year. Known recurring dates are then corrected post hoc when needed.

For each event, the LLM generates a structured JSON impact profile describing account-type-specific effects over multiple temporal windows. These include pre-event buildup, during-event multipliers, settlement waves, return phases, and accrual-specific mechanisms such as month-end cascades and close shifts. Count and amount effects are modeled separately rather than collapsed into a single multiplier. The LLM outputs are constrained by domain-specific prompts and validated against an explicit schema, which allows semantic richness without giving up control over the generator. A potential concern is that LLM-generated impact profiles could make the benchmark artificially easier if downstream models simply recovered simulator priors. We mitigate this by constraining LLM outputs through schema validation, bounded ranges, and downstream statistical sampling, while learning event embeddings from realized transaction residuals rather than directly from the generated profiles.

3.4 Event-aware transaction synthesis

Event-aware generation overlays the event calendar on top of the baseline account series. A key feature of this stage is per-account heterogeneity in event response. Each account is assigned latent sensitivity factors that determine how strongly it reacts to holidays, promotions, and event magnitude, so accounts within the same broad family need not respond identically to the same event. This is important because real financial systems rarely show perfectly uniform event effects across accounts.

The synthesizer also models delayed and asymmetric effects explicitly. Settlement lag shifts a fraction of deferred activity to future days, which is especially important for cash accounts. Return phases create negative post-event adjustments for revenue and receivable accounts. Overlapping events are combined using a dominant-event-wins strategy with hard caps, preventing unrealistic compounding spikes. Together, these design choices produce event-aware series with both immediate and lagged behavioral effects. The lag structures, return phases, and account-type-specific response magnitudes were chosen to match qualitative event-response patterns observed in real transaction data, such as delayed post-event settlement in cash accounts and concentrated near-event spikes followed by refund activity in receivable and revenue accounts.

3.5 Anomaly injection and labels

After generating clean event-aware data, we inject structured anomalies to create evaluation datasets for anomaly detection. By default, 30% of accounts are selected to receive anomalies, and each affected account receives 1–3 anomalies per year. Severity is sampled across minor, moderate, and major tiers.

The anomaly library spans three broad categories: point anomalies (magnitude spike, magnitude drop, sign reversal), contextual

Table 2: Main anomaly categories used in the benchmark.

Category	Example anomaly types	Why included
Point anomalies	Magnitude spike, magnitude drop, sign reversal	Test sensitivity to abrupt local deviations
Contextual anomalies	Holiday activity, missing expected event response	Test whether models use event context correctly
Temporal anomalies	Gradual drift, missing month-end close	Test detection of sustained structural deviations

anomalies (for example, unexpected holiday activity), and temporal anomalies (gradual drift and missing month-end close behavior). These anomalies were chosen so that some are detectable through generic deviation alone, while others require event awareness. For example, a large spike during Black Friday may be expected, whereas suspicious holiday activity or missing expected event response is anomalous only in context. Each anomaly is stored with structured metadata including anomaly type, account, date range, severity, event context, and injection parameters, which supports evaluation by anomaly family and account profile in addition to overall performance.

4 LEARNING IMPACT-DRIVEN EVENT EMBEDDINGS

The second stage of the framework learns compact event representations from the way events manifest in observed account activity. The goal of this stage is not merely to attach event information to the forecaster, but to learn shared representations of how recurring events appear in financial behavior after regular non-event structure has been removed. Rather than encoding an event only by name, type, or a binary active flag, we represent it through the residual response signature it induces across heterogeneous accounts relative to an estimated non-event baseline. These learned embeddings are later used as contextual covariates in downstream forecasting and anomaly detection.

4.1 Baseline estimation from observed series

A prerequisite for event representation learning is to isolate event-driven deviations from regular non-event structure. Although the synthetic benchmark provides access to the hidden baseline generator, using that hidden baseline directly would make the representation-learning stage unrealistically easy and would weaken its relevance to real deployment, where only observed time series and known event calendars are available. We therefore estimate a non-event reference directly from the observed event-aware daily series and compute event responses relative to that estimated baseline rather than to simulator ground truth.

In practice, we first construct complete daily panels for each account and mark short event-contamination windows around known events. The baseline is then fit using only uncontaminated periods. The exclusion window is deliberately local: its role is not to define the full event signature, but to prevent the most directly distorted event observations from leaking into the baseline estimator while still preserving enough clean data for stable fitting.

For most account profiles, the baseline combines a weekday-specific rolling-median component with a business-month-end adjustment. For accrual/month-end accounts, however, a separate branch is required because their dominant normal structure is a sharp close-window spike rather than a mild month-end uplift. The resulting residuals are then robustly normalized with MAD-based z-scores computed on clean periods only, producing a scale-comparable signal across accounts.

4.2 Event-centered response windows

Once the baseline-adjusted panel is constructed, we extract event-centered response windows around each event occurrence and align them in event-relative time. The response window is intentionally broader than the local exclusion window used during baseline fitting, because its purpose is to capture the full event signature, including pre-event buildup, event-core behavior, delayed settlement, and longer-tail post-event dynamics.

A natural first approach is to aggregate responses by exact relative day and account profile. In our setting, however, this proved too brittle. Accounts within the same profile do not all respond with the same intensity, and response timing is not perfectly synchronized: pre-event buildup can occur slightly earlier or later across accounts, settlement-related post-event spikes can vary in lag, and return effects can spread across several days. Exact day-level aggregation therefore tends to blur the event signature rather than sharpen it.

4.3 Profile-aware event response tensor

We first constructed a simple event tensor by aggregating event responses at the level of account profile and exact relative day. This representation preserved coarse differences across account profiles, but it discarded two forms of structure that proved important: within-profile heterogeneity and tolerance to small timing misalignment.

To address these limitations, we redesigned the representation as a richer event-response tensor that is profile-aware, distribution-aware, and timing-tolerant. Instead of exact-day medians only, event-relative time was grouped into asymmetric bins that are finer near the event and coarser farther away. For each event occurrence, account profile, and time bin, we computed summaries that capture central tendency, spread, and the prevalence of strong positive or negative responses across accounts, together with higher-level descriptors of temporal shape.

We further enriched this tensor with a small set of event-level features summarizing duration and overall response strength. The motivation was that the profile-aware tensor captured the temporal shape of an event well, but did not explicitly summarize whether the event was broad, sharp, or long-lasting at a global level. The final tensor therefore combines bin-level response summaries, temporal-shape descriptors, and event-level strength cues into a single representation.

4.4 Contrastive event embedding

The final event tensor is standardized, reduced with PCA, and passed to a contrastive encoder. The encoder is a multilayer perceptron that maps each event occurrence to a 16-dimensional embedding. The embedding size was set to 16 as a compact representation,

Table 3: Intrinsic evaluation of event representations.

Representation / model	Avg. same-name hits in top-5	Top-5 hit rate
Simple tensor + PCA	1.0166	70.00%
Simple tensor + autoencoder	1.0833	76.67%
Rich tensor + PCA	1.4167	90.00%
Rich tensor + contrastive encoder	1.9333	98.33%

chosen to balance expressiveness and compression. Positive pairs are defined as occurrences of the same named event across different years, reflecting the idea that recurring realizations of the same event should share a stable behavioral signature even when magnitude and account composition vary from year to year. Let $z_i = f_\theta(x_i)$ denote the embedding of event occurrence x_i, and let z_i^+ denote a positive pair for x_i. We train the encoder with an InfoNCE-style contrastive objective [16],

$$\mathcal{L}_{\text{InfoNCE}} = -\sum_i \log \frac{\exp(\text{sim}(z_i, z_i^+)/\tau)}{\exp(\text{sim}(z_i, z_i^+)/\tau) + \sum_{j \in \mathcal{N}(i)} \exp(\text{sim}(z_i, z_j)/\tau)},$$

where $\text{sim}(\cdot,\cdot)$ is cosine similarity, τ is the temperature, and $\mathcal{N}(i)$ denotes the negatives for event i in the minibatch. This encourages the embedding space to preserve recurring event structure while abstracting away year-specific variation and account-population noise.

4.5 Intrinsic evaluation and interpretation

We evaluated embedding quality using top-5 nearest-neighbor retrieval under cosine similarity, reporting the average number of same-name neighbors and the fraction of event occurrences with at least one same-name neighbor. These metrics test whether recurring realizations of the same event cluster together.

As shown in Table 3, the rich tensor substantially improves retrieval over the simple representation, while contrastive training achieves the strongest result, with a 98.33% top-5 hit rate. This indicates that profile-specific, distributional, and timing-tolerant structure provides most of the gain, with further improvement from contrastive learning.

5 EVENT-AWARE FORECASTING AND FORECAST-BASED ANOMALY DETECTION

5.1 Forecasting setup

We treat anomaly detection as event-aware probabilistic forecasting at the account-day level. For each account and day, the target is the daily signed transaction amount. The forecasting model receives historical context together with future-known covariates and outputs predictive quantiles over the forecast horizon. Anomalies are then defined downstream as deviations from the resulting event-conditioned forecast distribution.

We use Chronos-2 [1] as the main forecasting backbone. Chronos-2 is a pretrained covariate-aware probabilistic forecasting model that is used here in zero-shot mode, without fine-tuning on the synthetic benchmark. For each month of 2025, we construct a rolling forecast problem using a fixed 24-month context window ending on the day before the forecast month. To prevent temporal leakage, each forecast-period event is assigned a fixed embedding computed only from prior-year occurrences of the same recurring event; residuals from the forecast period are never used to construct its covariates. Forecasts are produced at quantile levels 0.02, 0.05, 0.10, 0.50, 0.90, 0.95, and 0.98, enabling both point and interval-based evaluation. The median forecast is used as the point prediction, and the corresponding quantile pairs define 80%, 90%, and 96% prediction intervals for the Chronos-based models.

5.2 Benchmark models and deployment constraints

For benchmarking, we prioritized forecasting methods that satisfy the deployment constraints of large-scale financial systems. Such systems may contain millions of heterogeneous account series, with new accounts appearing continuously, making repeated global retraining operationally impractical. We therefore selected Seasonal Naive, AutoETS, AutoTheta, and Prophet as lightweight baselines that can be fit or refreshed independently per series and applied immediately to newly added accounts [4, 6, 7, 14]. Prophet was also evaluated with a binary event regressor.

All Chronos-based models receive shared calendar covariates including cyclical encodings of day-of-week and month, together with weekend, month-start, and month-end indicators when applicable. For the Chronos-based models, we compare five event-covariate configurations: no event covariates, a binary event indicator, same-day learned event embeddings, lag-bucketed embeddings, and phase-aware embeddings. These differ in how event information is aligned in time. Same-day embeddings attach a single learned event vector to the event date itself. Lag-bucketed embeddings reuse the same event embedding across event-relative lag bins to capture delayed effects such as settlement. Phase-aware embeddings go further by using separate learned embeddings for four temporal phases of each event—pre, during, post-early, and post-late—so that different stages of the event response can be represented distinctly. For Prophet, we report both a no-event version and a version with a simple binary event regressor. Prophet produced 95% prediction intervals in our setup, which we use as an approximation to the 96% interval employed for the other forecasting models.

5.3 Forecast-based anomaly scores and final detector

The forecasting models provide both point forecasts and predictive quantiles, which define a family of anomaly scores. In addition to absolute and relative forecast error, we examine breaches of forecasted prediction intervals and normalized breach scores that scale interval violation by interval width. To account for practical relevance, we also compute a materiality ratio based on recent account scale.

Our final detector, `pred_flag_outside_96_material_duration`, begins with candidate days that fall outside the 96% prediction interval, applies a materiality threshold relative to recent account scale, and then adds duration-aware logic over runs of consecutive candidate days. Short isolated deviations are required to exceed a higher materiality threshold, while sustained runs are held to a lower threshold. Detector ablations reported in Appendix C.2

Table 4: Forecasting comparison across all benchmark models. We report WAPE as the main point forecast metric and interval score at the nominal 96% level as the main probabilistic metric. Prophet uses a 95% interval as an approximation. Lower is better for both metrics.

Model	WAPE	Interval score
Lag-bucketed embedding	0.2265	151,747
Same-day embedding	0.2308	145,645
Phase-aware embedding	0.2326	134,156
Binary event	0.2331	163,863
No event (Chronos)	0.2872	201,860
Prophet + binary event	0.3779	340,216
Prophet	0.3995	362,271
AutoTheta	0.5507	437,527
AutoETS	0.5851	507,381
Seasonal Naïve	0.6053	693,231

show a consistent precision-recall tradeoff across models: more conservative thresholding improves precision but reduces recall. We therefore report the materiality- and duration-aware detector as the main operating point because it achieved the strongest overall F1 across models.

5.4 Forecasting comparison across models

Table 4 compares the main forecasting models using WAPE and interval score. The learned-embedding Chronos variants clearly outperform both the no-event baseline and the traditional forecasting baselines. Lag-bucketed embeddings achieve the best WAPE overall, while phase-aware embeddings achieve the best interval score overall; same-day embeddings are a close second on both metrics. Binary-event conditioning also improves substantially over the no-event Chronos baseline, but remains weaker than the learned embedding variants.

Among the traditional methods, Prophet performs better than the lighter classical baselines, but still remains well behind the Chronos models with learned event representations. AutoTheta, AutoETS, and Seasonal Naïve are useful deployment-relevant baselines, but they are not competitive in this event-aware setting. These results suggest that the main gain does not come merely from using a stronger forecaster, but from providing it with structured event context.

5.5 Aggregate anomaly detection comparison

Table 5 reports aggregate anomaly-detection performance using the final detector. The strongest overall F1 is achieved by the lag-bucketed event embeddings, followed by phase-aware embeddings and binary event conditioning. Same-day embeddings remain competitive, but are weaker than the lag-bucketed variant on aggregate F1. All Chronos-based event-aware models clearly outperform the no-event baseline, and all of them substantially outperform Prophet and the lighter classical forecasting baselines.

This ranking is important because it shows that the gains are not simply due to using Chronos as a stronger forecaster. Even within the Chronos family, adding event structure improves downstream anomaly detection, and the best results come from the model variant whose event representation most closely matches delayed event-response dynamics.

Table 5: Aggregate anomaly-detection performance under the final detector `pred_flag_outside_96_material_duration`.

Model	Precision	Recall	F1
Lag-bucketed embedding	0.3513	0.6576	0.4580
Phase-aware embedding	0.3110	0.7217	0.4347
Binary event	0.2984	0.6221	0.4034
Same-day embedding	0.2652	0.7121	0.3865
No event (Chronos)	0.1659	0.5730	0.2573
Prophet + binary event	0.0524	0.5880	0.0962
Prophet	0.0524	0.5757	0.0961
AutoETS	0.0439	0.4025	0.0791
AutoTheta	0.0306	0.3438	0.0561
Seasonal Naïve	0.0166	0.2401	0.0311

Table 6: Profile-level anomaly detection under the final detector. We report F1 for the main Chronos-based models.

Account profile	Binary	Lag	No-event	Phase	Same-day
Accrual / month-end	0.0862	0.0195	0.1075	0.0205	0.0355
Cash	0.3364	0.6052	0.2835	0.5956	0.2350
Receivable	0.5897	0.6062	0.3734	0.5682	0.6302
Revenue	0.3898	0.4311	0.1879	0.3758	0.5295

Absolute precision remains moderate even for the strongest models, with the best aggregate precision reaching 0.3513 under the final detector. This reflects both the difficulty of the benchmark and the recall-sensitive operating point chosen for evaluation. In our setting, missing a true anomaly is often more costly than surfacing a manageable number of candidates for downstream review, so the detector is tuned as a screening-stage system rather than a highly conservative final decision rule.

5.6 Profile-level anomaly detection results

Because account profiles differ substantially in how strongly they are driven by external events, we focus the main profile-level analysis on the Chronos-based models. This breakdown is more informative than aggregate metrics alone, since accrual/month-end accounts are only weakly coupled to the modeled external event context and therefore act as a negative control.

Table 6 shows that the benefit of event embeddings depends strongly on account profile. Lag-bucketed embeddings perform best for cash accounts, which is consistent with delayed settlement behavior. Same-day embeddings perform best for receivable and revenue accounts, where event effects are concentrated during or near the event itself. No model materially improves accrual/month-end accounts, where all F1 values remain low and where external event context is least relevant. This profile-level pattern strengthens the interpretation that event embeddings help when their temporal structure matches the underlying business process, rather than by uniformly increasing model complexity.

5.7 Qualitative examples

Figures 2 and 3 illustrate how event timing affects financial account behavior. For the cash account, lag-bucketed conditioning anticipates delayed Black Friday / Cyber Monday settlement and avoids a false positive that the no-event model produces. For the receivable account, event-aware conditioning captures the near-event Prime

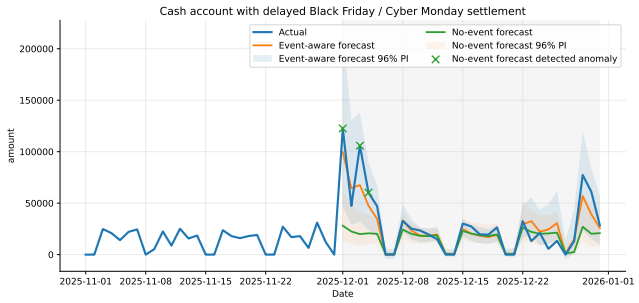

Figure 2: Cash account example. The no-event model flags the early-December settlement spike as anomalous, while the event-aware model anticipates the lagged response and avoids the false positive.

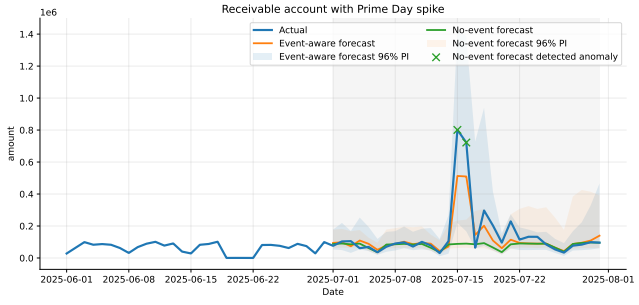

Figure 3: Receivable account example. The event-aware model captures the Prime Day spike as expected behavior, while the no-event model treats it as anomalous.

Day spike as expected promotional activity. In both cases, the embeddings improve detection by adjusting the forecast trajectory and uncertainty to the underlying financial process.

5.8 Interpretation

The results show that event-aware forecasting improves anomaly detection most clearly for account profiles whose dynamics are genuinely event-sensitive. The gains are not explained by adding event indicators alone: learned embeddings derived from residual response behavior outperform both no-event and binary-event baselines, as well as deployment-relevant traditional forecasting models. The profile-level results further show that temporal alignment matters: lag-bucketed embeddings are most effective for delayed-response cash accounts, while same-day embeddings are strongest for receivable and revenue accounts. In contrast, accrual/month-end accounts remain weakly affected by the modeled external events, suggesting that the embeddings help where event timing is operationally relevant rather than by uniformly increasing model complexity. A supplementary transfer experiment on unseen accounts with only 13 months of history showed the same qualitative pattern, with learned event representations continuing to outperform binary-event and no-event baselines (Appendix C.3).

6 DISCUSSION AND CONCLUSION

The results of this study support a broader view of anomaly detection in financial time series as a context-aware forecasting problem rather than a context-free outlier detection task. When recurring external events such as holidays and promotions are ignored, large but expected deviations are easily misclassified as anomalous. Incorporating known event context into the forecast distribution makes it possible to distinguish predictable event-driven variation from genuinely suspicious behavior, improving anomaly detection in the account families where such events are operationally meaningful.

More broadly, the findings suggest that event information is most useful when it is represented through realized behavioral impact rather than through event identity alone. In this sense, the main contribution is not simply the inclusion of event covariates, but a way of constructing event-aware context from observed transaction behavior and using it within a forecast-based anomaly-detection framework. The negative result for accrual/month-end accounts is also informative: it shows that the proposed event representations do not artificially improve every series, but mainly help where external event timing is genuinely relevant.

This study has several limitations. First, all quantitative evaluation is conducted on a synthetic benchmark rather than on labeled production data. Although the simulator was calibrated against empirical signatures from real financial transaction streams, any synthetic benchmark necessarily reflects modeling assumptions and may omit important real-world phenomena such as regime shifts, noisy metadata, changing business processes, incomplete event calendars, or unmodeled operational dependencies. Although policy constraints prevent us from reporting results on the underlying internal data, parallel internal benchmarking showed comparable qualitative improvements, which supports the practical relevance of the synthetic findings. Second, the benchmark is instantiated in a U.S. retail and financial-operations setting, so the event mix and account behavior may differ in other regions or industries. Third, the current framework assumes that the event calendar is known in advance and reasonably clean, whereas real deployments may face missing, delayed, or ambiguous event information.

A natural next step is retrospective validation on real account-level data, including descriptive comparison of learned event-response structure and anomaly-review studies based on analyst or audit outcomes. More broadly, future work could examine robustness to incomplete event calendars, changing account populations, and weaker or noisier event metadata.

Overall, the paper supports the claim that anomaly detection in financial time series is better framed as a forecasting problem in which known events influence both expected behavior and predictive uncertainty. Learning event representations from realized impact, rather than from event identity alone, provides a practical way to incorporate that context and reduce false positives in event-sensitive settings. In large-scale financial environments, this matters because reducing predictable event-driven alerts can improve the efficiency of downstream review workflows without relying on expensive globally retrained models.

REFERENCES

[1] Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan

Zhang, et al. 2025. Chronos-2: From Univariate to Universal Forecasting. *arXiv preprint arXiv:2510.15821* (2025).

[2] 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. 2024. Chronos: Learning the Language of Time Series. *arXiv preprint arXiv:2403.07815* (2024).

[3] Sebastian Pineda Arango, Pedro Mercado, Shubham Kapoor, Abdul Fatir Ansari, Lorenzo Stella, Huibin Shen, Hugo Senetaire, Caner Turkmen, Oleksandr Shchur, Danielle C. Maddix, et al. 2025. ChronosX: Adapting Pretrained Time Series Models with Exogenous Variables. *arXiv preprint arXiv:2503.12107* (2025).

[4] Vassilios Assimakopoulos and Konstantinos Nikolopoulos. 2000. The theta model: A decomposition approach to forecasting. *International Journal of Forecasting* 16, 4 (2000), 521–530.

[5] Stéphane Crépey et al. 2022. Anomaly Detection on Financial Time Series by Principal Component Analysis and Neural Networks. *arXiv preprint arXiv:2209.11686* (2022).

[6] Rob J. Hyndman and George Athanasopoulos. 2021. *Forecasting: Principles and Practice* (3 ed.). OTexts.

[7] Rob J. Hyndman, Anne B. Koehler, Ralph D. Snyder, and Simone Grose. 2002. A State Space Framework for Automatic Forecasting Using Exponential Smoothing Methods. *International Journal of Forecasting* 18, 3 (2002), 439–454. https://doi.org/10.1016/S0169-2070(01)00110-8

[8] Sebastian Kiefer and Günter Pesch. 2021. Unsupervised Anomaly Detection for Financial Auditing with Model-Agnostic Explanations. In *KI 2021: Advances in Artificial Intelligence (Lecture Notes in Computer Science, Vol. 12873)*. Springer, 291–308. https://doi.org/10.1007/978-3-030-87626-5_22

[9] Alexander Kogan, Michael G. Alles, Miklos A. Vasarhelyi, and Jia Wu. 2014. Design and Evaluation of a Continuous Data Level Auditing System. *Auditing: A Journal of Practice & Theory* 33, 4 (2014), 221–246. https://doi.org/10.2308/ajpt-50844

[10] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 413–422. https://doi.org/10.1109/ICDM.2008.17

[11] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-Based Encoder-Decoder for Multi-Sensor Anomaly Detection. https://doi.org/10.48550/arXiv.1607.00148 arXiv:1607.00148 [cs.LG]

[12] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. 2015. Long Short Term Memory Networks for Anomaly Detection in Time Series. In *Proceedings of the 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*.

[13] Marco Schreyer, Timur Sattarov, Christian Schulze, Bernd Reimer, and Damian Borth. 2019. Detection of Accounting Anomalies in the Latent Space using Adversarial Autoencoder Neural Networks. *arXiv preprint arXiv:1908.00734* (2019).

[14] Sean J. Taylor and Benjamin Letham. 2018. Forecasting at Scale. *The American Statistician* 72, 1 (2018), 37–45.

[15] Sutapat Thiprungsri and Miklos A. Vasarhelyi. 2011. Cluster Analysis for Anomaly Detection in Accounting Data: An Audit Approach. *The International Journal of Digital Accounting Research* 11, 17 (2011), 69–84. https://doi.org/10.4192/1577-8517-v11_4

[16] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748* (2018).

[17] Y. Wang, H. Wu, J. Dong, G. Qin, H. Zhang, Y. Liu, Y. Qiu, J. Wang, and M. Long. 2024. TimeXer: Empowering Transformers for Time Series Forecasting with Exogenous Variables. In *Advances in Neural Information Processing Systems*.

[18] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. 2022. CoST: Contrastive Learning of Disentangled Seasonal-Trend Representations for Time Series Forecasting. *arXiv preprint arXiv:2202.01575* (2022).

[19] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. 2022. ETSformer: Exponential Smoothing Transformers for Time-series Forecasting. *arXiv preprint arXiv:2202.01381* (2022).

[20] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *International Conference on Learning Representations*.

[21] Yosuke Yamaguchi, Issei Suemitsu, and Wenpeng Wei. 2025. CITRAS: Covariate-Informed Transformer for Time Series Forecasting. *arXiv preprint arXiv:2503.24007* (2025).

[22] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. 2021. TS2Vec: Towards Universal Representation of Time Series. *arXiv preprint arXiv:2106.10466* (2021).

A ADDITIONAL SYNTHETIC BENCHMARK DETAILS

This appendix provides implementation-level details for the synthetic benchmark introduced in Section 1. It focuses on the concrete benchmark instantiation used in the experiments, the parameterization of the account simulator, the constrained LLM-assisted event generation pipeline, and the anomaly injection setup. The goal is to supplement the main paper with technical detail rather than to restate the high-level benchmark design.

A.1 Benchmark instantiation used in this paper

The experimental benchmark used in this paper spans three years of daily data (2023–2025) and contains 1,000 accounts. Accounts are drawn from four behavioral families: cash/banking, receivables, accrual/month-end, and revenue. The generator supports configurable profile mixtures; in the default implementation, the profile mix is uniform across the four families unless otherwise specified. The current benchmark is instantiated for a U.S. retail and financial-operations context, which determines the holiday calendar and the recurring commercial events used in the event layer.

Although the current experiments use a U.S. event environment, the framework itself is not U.S.-specific. The event calendar, prompt templates, known-date anchors, and schema constraints can be adapted to other geographies and holiday systems by changing the region-specific event rules and prompt context.

A.2 Baseline generator parameterization

The baseline generator is a profile-driven probabilistic simulator. Each account receives a profile-specific configuration, after which additional per-account heterogeneity is introduced through sampling. The core account families differ in cadence, activity calendar, seasonality structure, and amount scaling.

At generation time, each account is assigned an activity mode, a weekly template, a monthly profile, a regularity score, a trend direction and strength, and account-specific amount scaling and distribution parameters.

Daily transaction generation then proceeds through the following sequence: (i) determine whether the account is active on the day using an activity gate; (ii) compute expected daily count by combining base mean with trend, weekly, and monthly effects; (iii) sample realized count from a negative binomial distribution; (iv) sample transaction magnitudes from a lognormal distribution; and (v) apply sign policy and assign random timestamps within the day.

Table 7: Profile-level baseline parameterization summary.

Account family	Cadence / activity style	Weekly / monthly structure
Cash	Mostly business-day activity, near-zero weekends	Monday batching, flat weekdays, occasional Friday spike; modest month-end uplift in some accounts
Receivables	Mostly all-day activity, moderate weekend suppression	Mild weekend suppression, some month-end uplift
Accrual / month-end	Month-end concentrated with optional off-cycle drizzle	No true weekly pattern; spike in final two days of month
Revenue	Mix of all-day and business-day activity	Flat or weekend-high weekly structure; mild month-end uplift

The profile library also specifies constants such as month-end activity probability, off-cycle mean scaling, regularity effects on dispersion, and amount-noise scaling. These are implementation details that were chosen to preserve behavioral differences across families while maintaining controllable heterogeneity.

A.3 Calibration to production data

Although all downstream experiments in this paper are performed on synthetic data, the simulator parameters were calibrated against empirical analysis of production financial transaction data from the target application setting. The calibration followed a two-stage process.

In the first stage, we extracted aggregate statistical signatures and qualitative temporal response patterns from real account-level time series across the four account families used in the benchmark: cash/banking, receivables, accrual/month-end, and revenue. These signatures were used to anchor the baseline generator and the event-impact schema. For cash accounts, the empirical data showed near-zero holiday activity, next-business-day backlog effects, pre-holiday cash drawdown, and post-holiday settlement spikes with delays of several days. Receivable accounts showed steady buildup, processor-settlement cycles, Friday–Monday effects, strong event-day promotional spikes, and later refund-related reversals. Accrual accounts showed extreme month-end concentration, amplified quarter-end and year-end closes, and close-window shifts around holidays. Revenue accounts showed high daily volume, clear weekday effects, holiday suppression, strong pre-holiday elevation, large promotional surges, and post-event return activity.

These empirical signatures informed the constraint ranges embedded in the event-impact schema, including multiplier ceilings, return-fraction bounds, settlement-lag ranges, and magnitude-tier guidance, as well as the baseline profile parameters such as activity-mode probabilities, weekly template weights, month-end spike magnitudes, and dispersion ranges.

In the second stage, the generator was iteratively refined through qualitative validation. Synthetic time series produced under candidate parameter settings were visually compared against real account-level series to verify that they reproduced the characteristic temporal

structures observed in practice, including weekly business-day cycles, month-end concentration, event-driven suppression and recovery dynamics, settlement timing, and within-family heterogeneity in responsiveness. This process was used to tune parameters such as the Beta-distribution shapes controlling per-account event sensitivity, the count-dispersion ranges, and the activity-mode probabilities governing weekend-to-weekday behavior.

This calibration does not imply that the benchmark fully reproduces every property of a production ledger, nor do we claim that synthetic success alone guarantees production performance. Its purpose is narrower: to ensure that the simulated benchmark is anchored to realistic statistical regimes and event-response behaviors while remaining fully synthetic and fully labeled.

A.4 Event calendar generation and known-date anchoring

The event calendar is generated through a hybrid pipeline that combines deterministic rules with LLM-generated structure. Known recurring events are computed explicitly from calendar logic. For example, Thanksgiving is defined as the fourth Thursday of November, Black Friday as the following day, Cyber Monday as the Monday after Black Friday, and Memorial Day as the last Monday of May. These known-date anchors are applied as a post-processing correction step after LLM generation, so recurring events always land on the correct date even if the model makes a calendar error.

The LLM-generated calendar produces a yearly list of events with fields such as event ID, event type, event name, start and end dates, magnitude tier, category, and short event description. For the current benchmark, the prompt requests major U.S. bank holidays, flagship retail events such as Prime Day, Black Friday, and Cyber Monday, and several minor or moderate promotional events distributed across the year.

A.5 LLM-generated impact profiles under schema constraints

For each event in the calendar, the LLM generates a structured JSON impact profile describing how the event affects different account types. The output is constrained by a detailed system prompt encoding financial-domain rules, an explicit JSON schema, parse and validation checks, and retry logic on invalid output.

The schema supports account-type-specific fields including `pre_event_phases`, `during_event`, `settlement_waves`, `return_phases`, `settlement_lag`, `month_end_cascade`, `accrual_close_shift`, and `amount_adjustment`.

A key design decision is that count and amount multipliers are modeled independently. This avoids unrealistic simplifications in which event-driven transaction volume and transaction size are assumed to move together. The prompt explicitly discourages identical count and amount multipliers and encodes account-specific rules such as restricted return mechanisms for account types where returns are operationally meaningful.

A.6 Event multiplier engine and event-aware synthesis

The LLM does not directly emit per-day transaction series. Instead, its impact profiles define bounded ranges and temporal windows that are later converted into day-level event multipliers by a statistical engine. This engine samples concrete scalar multipliers from the LLM-provided ranges and constructs per-day event effects for each account type.

The multiplier engine supports linear, exponential, and step-style ramps; pre-event, during-event, and post-event windows; settlement waves; return accumulators; and month-end cascade effects for accrual accounts.

When events overlap, aggregation is not handled by unconstrained multiplication. Instead, the calendar uses a dominant-event-wins rule that selects the multiplier with the largest deviation from baseline, and hard caps are applied to count and amount multipliers to prevent unrealistic explosive spikes.

The event-aware synthesizer then overlays these multipliers on top of the baseline generator. It maintains lag and return accumulators at the account level, so deferred settlement volume and return effects are released on future days rather than being approximated through static post-event scaling.

A.7 Per-account sensitivity model

To prevent all accounts within a family from responding identically to the same event, the generator assigns each account three latent sensitivity factors: holiday sensitivity, promotional sensitivity, and magnitude sensitivity.

These factors are sampled from profile-specific Beta distributions. For example, cash accounts are highly sensitive to holidays but much less directly sensitive to promotions, revenue and receivable accounts are highly sensitive to promotional events, and accrual accounts are relatively insensitive to external event timing but still allow magnitude-related changes.

The effective multiplier applied to a given account-event pair is obtained by scaling the account-type-level multiplier using the relevant sensitivity axis and the event's magnitude tier. This introduces realistic within-profile heterogeneity without destroying the profile-level behavioral signatures.

A.8 Anomaly injection setup

Anomalies are injected only after clean event-aware data has been generated. In the default anomaly configuration, 30% of accounts receive at least one anomaly, each affected account receives 1–3 anomalies per year, and anomaly severity is sampled across minor, moderate, and major with roughly balanced weights.

The implemented anomaly registry includes `magnitude_spike`, `magnitude_drop`, `sign_reversal`, `gradual_drift`, `missing_month_end`, and `missing_event_response`. The injection logic is account-type-aware. For example, accrual accounts receive magnitude spikes and missing month-end anomalies, cash accounts receive spikes, drops, and drift, and revenue and receivable accounts can receive spikes, drops, sign reversals, and event-conditioned anomalies.

Each injected anomaly is exported as a structured label record with anomaly ID, type, account, account type, date range, severity, optional event context, description, and raw injection parameters. This label schema is used directly in downstream evaluation.

A.9 LLM prompting, validation, and control

The benchmark uses an LLM in two controlled roles: yearly event calendar generation and per-event impact profile generation. In both cases, the LLM operates inside a constrained generation pipeline rather than as a free-form text generator. Deterministic calendar rules, structured output requirements, bounded parameter ranges, and downstream statistical sampling preserve control while still allowing semantic variety in the generated events and impact profiles.

For event calendar generation, the prompt explicitly requests a U.S. retail and financial-operations calendar and requires the model to return a structured JSON array of events. A representative excerpt from the event-calendar system prompt is:

> "You are an expert in US retail and financial operations calendars. Your task is to generate a realistic event calendar for a given year for use in synthetic accounting transaction data generation."

For impact-profile generation, the prompt encodes domain-specific constraints on how different account types should respond to events. A representative excerpt from the impact-profile system prompt is:

> "You are a financial data modeling expert. Your task is to generate realistic impact profiles for financial transaction events. Each profile describes how a specific event affects different account types in terms of transaction volume (count) and transaction size (amount)."

The same prompt then imposes additional rules, for example that count and amount multipliers must be treated as independent axes, major events may require multi-phase pre-event buildup, and settlement waves and return phases should only be used where operationally appropriate. Generated outputs are checked against the required structure, and known recurring event dates are corrected using deterministic calendar rules when needed.

The purpose of this design is not to rely on unconstrained LLM generation for realism. Instead, the LLM contributes semantic variety and plausible event structure, while the simulator preserves control through bounded ranges, validation, correction, and downstream statistical sampling.

A.10 Why these details are kept in the appendix

These implementation details are included here because they support trust and reproducibility, but they are intentionally kept out of the main text to avoid interrupting the paper's central narrative. The main paper focuses on the role of the synthetic benchmark in enabling controlled evaluation. This appendix records the concrete design choices that make that benchmark realistic, constrained, and auditable.

B ADDITIONAL DETAILS ON EVENT EMBEDDING CONSTRUCTION

B.1 Exact tensor construction

The event-response tensor is constructed from event-centered windows spanning 14 days before event start to 30 days after event start. In the final representation, event-relative time is grouped into asymmetric bins that are narrower near the event core and wider farther away. For each event occurrence, account profile, and time bin, the tensor includes distributional summaries such as central tendency, spread, quantiles, fractions of strongly positive or negative responses, and signed positive and negative mass statistics. In addition, profile-level temporal-shape descriptors summarize how response mass is distributed across time, including peak response, total response mass, temporal center and spread, and broader region-level summaries over pre-event, event-core, and post-event phases. The final tensor is further enriched with event-level strength features including duration, global response strength, impact per day, number of active profiles, and peak absolute response.

B.2 Contrastive training details

The final event tensor is standardized and reduced with PCA before contrastive training. The encoder is a multilayer perceptron with a 16-dimensional output embedding. We used 16 dimensions as a compact default: exploratory PCA indicated that the first 16 components retained about 75% of the variance in the event-response features, suggesting a reasonable tradeoff between compression and representational

Table 8: Extended forecasting metrics across all benchmark models. Lower is better for MAE, RMSE, WAPE, and interval score. Coverage is reported at the nominal 96% level for the Chronos-based models and 95% for Prophet.

Model	MAE	RMSE	WAPE	Coverage	Interval score
Lag-bucketed embedding	14,174.19	51,892.39	0.2265	0.9856	151,747
Same-day embedding	14,656.26	53,644.57	0.2308	0.9837	145,645
Phase-aware embedding	14,669.67	54,909.43	0.2326	0.9849	134,156
Binary event	15,059.53	58,962.80	0.2331	0.9807	163,863
No event (Chronos)	18,382.05	72,165.44	0.2872	0.9747	201,860
Prophet + binary event	23,858.36	87,884.14	0.3779	0.9625	340,216
Prophet	25,403.03	92,414.57	0.3995	0.9658	362,271
AutoTheta	34,909.20	116,259.02	0.5507	0.9719	437,527
AutoETS	37,583.74	142,917.32	0.5851	0.9729	507,381
Seasonal Naïve	38,370.59	146,488.00	0.6053	0.9641	693,231

capacity. Positive pairs are defined as occurrences of the same named event across different years. Training uses a symmetric InfoNCE objective [16] with temperature 0.15, batch size 32, learning rate 10^{-3}, weight decay 10^{-4}, and early stopping.

C ADDITIONAL FORECASTING AND ANOMALY DETECTION RESULTS

C.1 Extended forecasting metrics

Table 8 reports the full forecasting metrics for all benchmark models. In the main paper, we focus on WAPE and interval score because they align most directly with the forecast-based anomaly-detection objective. The broader metrics tell a consistent story: the Chronos models with learned event representations dominate the deployment-relevant traditional baselines, while Prophet improves over the lighter classical baselines but remains substantially behind the event-embedding models. At the same time, the strongest point-forecast metrics and the strongest downstream anomaly-detection results are not perfectly aligned, suggesting that useful event conditioning depends on uncertainty calibration as well as point accuracy.

C.2 Detector ablation summary

Table 9 reports the detector ablation for each forecasting model. The main pattern is consistent across models: moving from a raw interval-breach rule to materiality-aware filtering improves precision, and adding duration-aware filtering yields the strongest final F1. This supports the use of the final detector `pred_flag_outside_96_material_duration` in the main paper.

C.3 Transfer to unseen accounts with limited history

To test whether the learned event representations transfer beyond the accounts used during embedding construction, we ran an additional experiment on a disjoint set of 1,000 accounts that were not used to learn the event embeddings and that had only 13 months of training history available. This setting is meant to approximate newly added accounts in production, where long account-specific histories are often unavailable. The evaluation accounts shared the same event environment as the main benchmark, but the embeddings themselves were learned on a separate account set.

Table 10 reports the forecasting results in this reduced-history transfer setting. The qualitative pattern remains consistent with the main experiments: all event-aware variants outperform both the binary-event and no-event baselines. Same-day embeddings achieve the best WAPE, while phase-aware embeddings achieve the best interval score. These results support the claim that the learned event representations capture transferable event-response structure rather than merely memorizing the accounts used during embedding learning.

Table 11 reports downstream anomaly-detection performance under the same transfer setting. Although absolute anomaly-detection performance is lower than in the main experiment, the relative advantage of learned event representations remains strong. All three learned embedding variants outperform both the binary-event and no-event baselines on F1, and the gap relative to the no-event baseline is even larger than in the main experiment. This suggests that transferable event-response representations are especially valuable when account-specific history is limited. The transfer setting also changes the relative ranking of temporal alignments. While lag-bucketed embeddings achieve the strongest aggregate anomaly F1 in the main experiment, phase-aware embeddings perform best on unseen accounts with limited history. We interpret this as a robustness effect: lag-bucketed alignment is especially effective when sufficient account history is available to support delayed-settlement dynamics, whereas phase-aware covariates provide a coarser event-lifecycle structure that transfers better when account-specific history is limited.

C.4 Profile-level precision, recall, and F1

Table 12 reports the full precision, recall, and F1 breakdown by account profile for the main Chronos-based models. In particular, the table makes clear that gains from event-aware conditioning are concentrated in cash, receivable, and revenue profiles, while accrual/month-end accounts remain relatively insensitive to the modeled event context.

Table 9: Detector ablation across Chronos-based forecasting models. The final 96% materiality- and duration-aware detector achieves the strongest F1 for each model.

Model	Detector	Precision	Recall	F1
Binary event	Outside 90% PI	0.0198	0.8840	0.0388
Binary event	Outside 90% PI + materiality	0.0464	0.8117	0.0878
Binary event	Outside 96% PI	0.0641	0.6617	0.1169
Binary event	Outside 96% PI + materiality	0.1079	0.6357	0.1845
Binary event	Outside 96% PI + materiality + duration	0.2984	0.6221	**0.4034**
Lag-bucketed embedding	Outside 90% PI	0.0181	0.8772	0.0355
Lag-bucketed embedding	Outside 90% PI + materiality	0.0424	0.8090	0.0806
Lag-bucketed embedding	Outside 96% PI	0.0878	0.6958	0.1559
Lag-bucketed embedding	Outside 96% PI + materiality	0.1323	0.6767	0.2213
Lag-bucketed embedding	Outside 96% PI + materiality + duration	0.3513	0.6576	**0.4580**
No event	Outside 90% PI	0.0152	0.8158	0.0299
No event	Outside 90% PI + materiality	0.0300	0.7667	0.0577
No event	Outside 96% PI	0.0466	0.6180	0.0866
No event	Outside 96% PI + materiality	0.0714	0.5948	0.1274
No event	Outside 96% PI + materiality + duration	0.1659	0.5730	**0.2573**
Phase-aware embedding	Outside 90% PI	0.0183	0.8936	0.0360
Phase-aware embedding	Outside 90% PI + materiality	0.0400	0.8131	0.0763
Phase-aware embedding	Outside 96% PI	0.0901	0.7572	0.1610
Phase-aware embedding	Outside 96% PI + materiality	0.1315	0.7326	0.2230
Phase-aware embedding	Outside 96% PI + materiality + duration	0.3110	0.7217	**0.4347**
Same-day embedding	Outside 90% PI	0.0188	0.9127	0.0368
Same-day embedding	Outside 90% PI + materiality	0.0430	0.8295	0.0817
Same-day embedding	Outside 96% PI	0.0854	0.7626	0.1536
Same-day embedding	Outside 96% PI + materiality	0.1240	0.7312	0.2120
Same-day embedding	Outside 96% PI + materiality + duration	0.2652	0.7121	**0.3865**

Table 10: Forecasting on unseen accounts with limited history. The event embeddings were learned on a disjoint account set, and the evaluation accounts had only 13 months of training history available. Lower is better for both metrics.

Model	WAPE	Interval score
Same-day embedding	0.2458	190,127
Phase-aware embedding	0.2499	167,443
Lag-bucketed embedding	0.2502	203,115
Binary event	0.2596	229,981
No event (Chronos)	0.3233	289,553

Table 11: Aggregate anomaly-detection performance on unseen accounts with limited history under the final detector `pred_flag_outside_96_material_duration`. The event embeddings were learned on a disjoint account set, and the evaluation accounts had only 13 months of training history available.

Model	Precision	Recall	F1
Phase-aware embedding	0.2768	0.6480	0.3876
Lag-bucketed embedding	0.2509	0.5948	0.3529
Same-day embedding	0.1957	0.6876	0.3047
Binary event	0.1596	0.6139	0.2534
No event (Chronos)	0.0870	0.5648	0.1508

Table 12: Profile-level precision, recall, and F1 under the final detector for the main Chronos-based models.

Account profile	Metric	Binary	Lag-bucketed	No-event	Phase-aware	Same-day
accrual_month_end	Precision	0.045455	0.009934	0.057471	0.010453	0.018405
	Recall	0.833333	0.500000	0.833333	0.500000	0.500000
	F1	0.086207	0.019481	0.107527	0.020478	0.035503
cash_accounts	Precision	0.217456	0.526119	0.174941	0.494983	0.139961
	Recall	0.742424	0.712121	0.747475	0.747475	0.732323
	F1	0.336384	0.605150	0.283525	0.595573	0.235008
receivable_accounts	Precision	0.562712	0.560127	0.296703	0.453333	0.567164
	Recall	0.619403	0.660448	0.503731	0.761194	0.708955
	F1	0.589698	0.606164	0.373444	0.568245	0.630182
revenue_accounts	Precision	0.308725	0.331276	0.115385	0.261654	0.423963
	Recall	0.528736	0.616858	0.505747	0.666667	0.704981
	F1	0.389831	0.431058	0.187900	0.375810	0.529496