RunTime: Distributional Transformers for Irregular Event Sequences

A causal Transformer framework for distributional regression on event trajectories

Yael S. Elmatad

January 2026

The problem

Many real-world systems are **event-driven**:

- events arrive at **irregular intervals**
- each event has context (environment/state) and an outcome
- uncertainty matters (tails drive decisions)

Typical regression flattens history into a table and predicts a point estimate, losing cadence and uncertainty.

Key idea (one sentence)

Represent each event as a fixed token block, quantize continuous variables into bins, treat **time deltas as tokens**, and train a **causal Transformer** to predict a **full PDF** over outcomes (not just a point).

RunTime at a glance

- **Input**: tokenized trajectory (stride-based blocks) + irregular time tokens
- **Model**: masked self-attention (causal)
- **Output**: softmax distribution over discretized outcome bins
- **Training**: cross-entropy with **seconds-aware Gaussian smoothing** targets

Why predict a PDF (not just a point estimate)?

Point estimates hide risk:

- multimodality and heavy tails collapse into a single number
- confidence becomes invisible

RunTime outputs a **probability density function (PDF)** over discretized bins, enabling:

- uncertainty-aware decisions
- median/mean/mode point predictions as summaries
- auditability via attention/activation inspection

The 11-token event block (grammar)

Each event is encoded as a fixed stride (11 tokens), mixing context, cadence, and target:

- demographics: `age_*`, `gen_*`
- context: `cond_*`, `hum_*`, `temp_*`, `feels_*`, `wind_*`
- distance: `distance_name_token_*`
- cadence: `d_next_*` (time since previous), `d_fin_*` (time to target)
- outcome: `pace_*` (the supervised target token)

Example “sentence” (two-event trajectory)

Event 1 (history):

```
[age_35, gen_M, cond_Clear, hum_45, temp_55, feels_55, wind_5,
distance_name_token_10_kilometers, d_next_12, d_fin_12, pace_115]
```

Event 2 (target context, pace is predicted):

```
[age_35, gen_M, cond_Rain, hum_85, temp_48, feels_42, wind_15,
distance_name_token_half_marathon, d_next_0, d_fin_0, pace_?]
```

The model conditions on the full token sequence and predicts a **distribution** over the final pace token.

Week-delta tokens: injecting cadence into the sequence

Instead of a single numeric feature, time is represented as tokens:

- `d_next`: time between adjacent events (cadence / recency)
- `d_fin`: time remaining to the final target event (horizon)

This makes irregularity explicit and lets attention learn non-linear decay / momentum effects.

Missing data, imputation, and survival-style time-to-event

- Many forecasters assume a **regular grid**; irregularity is handled via **imputation/resampling**, which can **smear signal** and hide meaningful gaps.
- RunTime treats **irregular event time** as first-class input (time tokens), so "no observation" can remain a **gap** rather than a fabricated value.
- **Missingness tokens generalize beyond time**: you can represent absent covariates explicitly (e.g., `temp_missing`, `humidity_missing`) instead of imputing continuous values.
  - This plays nicely with mixed representations: keep continuous-valued features continuous when present, but include a dedicated `*_missing` token/state when absent.
- For **time-to-next-event** tasks (survival/churn/retention), the discrete grammar can represent informative missingness:
  - `time_censored`: no future event within the observation window
  - `time_missing`: next-event timing unknown but the entity remains active
- This enables **distributional** time-to-event prediction (e.g., "time-to-next-admission" PDFs) with explicit handling of censoring.

Balanced quantization (binning strategy)

Continuous variables are discretized into bins with a simple goal: **maximize usable signal per token**.

- use **balanced** (approximately equal-mass) bins rather than equal-width bins
- yields higher entropy targets and avoids extremely sparse classes
- bin widths adapt: narrow where data is dense, wider where data is sparse

The model predicts a PDF over pace bins; bin medians support mean/median/mode summaries.

Quantization as regularization (and minimal tuning)

Quantization also acts as a structural regularizer:

- reduces sensitivity to micro-noise in seconds-level outcomes
- encourages learning transitions between meaningful states

Practical note: the core results here were achieved with **minimal optimization**:

- bin boundaries were generated once (not heavily tuned)
- the value comes from the architecture + training objective, not brittle heuristics

Point estimate vs PDF: what changes in practice?

A point estimate answers: "what number should I predict?"

A PDF answers: "how much probability mass lies in each outcome region?"

That enables:

- tail-aware decisions (thresholds, risk budgets, intervention timing)
- calibrated confidence (wide vs narrow PDFs)
- multiple decision-relevant summaries (mean/median/mode/quantiles)

Monte Carlo trajectory simulation (cone of uncertainty)

Once the model outputs a PDF at each step, you can simulate plausible futures by **recursive sampling**.

At step $t+1$ sample:

$$\tilde{y}_{t+1} \sim P(y \mid X_{\leq t})$$

Repeat this rollout for M samples to get a **family of trajectories** $\{\mathcal{T}^{(m)}\}_{m=1}^{M}$, which forms a cone of uncertainty over long horizons.

What you can answer:

- 80% interval (or any quantile band) for the state at a future horizon
- probability of crossing a threshold (tail mass) within the next k steps

Token order is a hidden hyperparameter (generative rollouts)

Auto-regressive generation defines a factorization:

$$P(x_{1:n}) = \prod_{i=1}^{n} P(x_i \mid x_{<i})$$

- If you generate a full future event block token-by-token, the **token ordering** implicitly chooses which conditionals you model (e.g., $P(\text{temp} \mid \text{dist}, \ldots)$ vs $P(\text{dist} \mid \text{temp}, \ldots)$).
- To avoid incoherent "stories" in rollouts:
  - **Conditional generation (recommended)**: treat future covariates (distance, weather) as given; only sample the target (pace).
  - **Joint generation (if needed)**: decide what is exogenous vs endogenous and design/test token order as an explicit modeling choice.

Seconds-aware Gaussian smoothing (targets)

Instead of "hard" one-hot targets, construct a soft target distribution T_i by integrating a Gaussian kernel over each bin interval:

$$T_i = \int_{b_i^{start}}^{b_i^{end}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x - y_{true}}{\sigma}\right)^2} dx$$

This yields:

- **ordinal awareness**: near misses are rewarded
- smoother training dynamics + better calibrated PDFs

Benchmark setup (high level)

- evaluation split is **entity-disjoint** (unseen runners)
- report MAE in **seconds per mile**
- compare against simple baselines + XGBoost tabular regressor

XGBoost baseline (design choices): engineered pace stats (mean/EMA/min/max/volatility), cadence features, binned weather/context features, **continuous** distance_miles from distance token, **one-hot** conditions, early stopping; artifacts written to an output directory.

Continuous vs discretized: "resolution disadvantage"

- **XGBoost** regresses directly to the **continuous** target (pace in seconds).
- **RunTime** predicts a **PDF over discretized pace bins** ($\sim 270+$ classes); point estimates come from mean/median/mode over bin medians.
- **Chronos-2 comparison**: values remain **discretized tokens**, while irregular gaps are represented via **continuous time embeddings** (normalized time deltas).
- Discretization is both a **constraint** and a **regularizer**: it reduces sensitivity to micro-noise and helps avoid collapsing to a safe average.
- Despite this "resolution disadvantage", RunTime remains competitive with (or outperforms) the tuned tabular baseline.

Baseline signal hierarchy (what matters most)

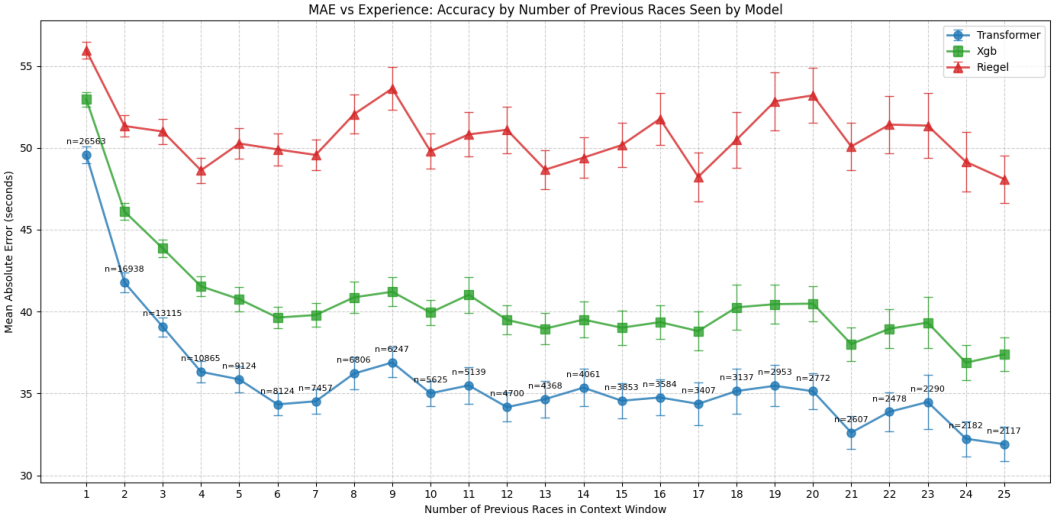
- In the XGBoost baseline, **historical performance** (EMA + distance-specific averages) dominates feature importance ($> 75\%$).
- Environmental features (e.g., temperature/humidity) tend to contribute little ($< 2\%$) relative to history.
- This motivates RunTime's core thesis: modeling **system rhythm** (history + cadence) is the main driver of accuracy.

Results (runner-disjoint MAE)

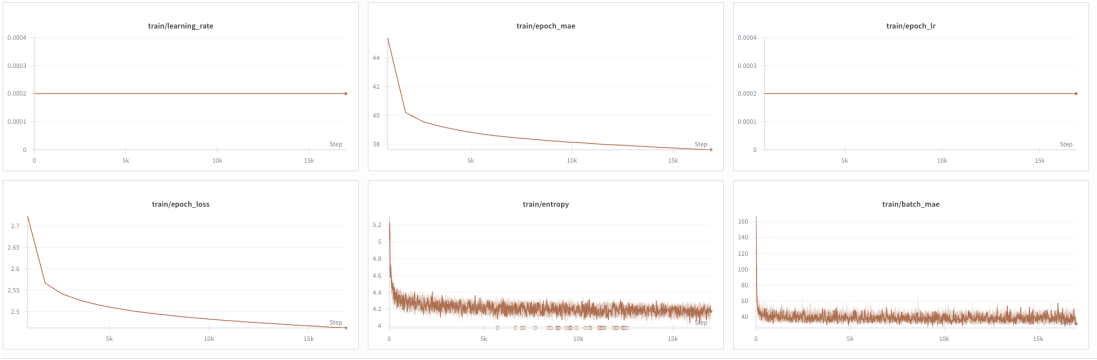
Model	MAE (s)
Naive Mean	54.19
Last Race Pace	61.31
Riegel Formula	50.76
XGBoost	40.94
Transformer (Mean)	37.67
Transformer (Median)	**37.10**
Transformer (Mode)	38.64

Improvement vs XGBoost (median): $1 - 37.10/40.94 \approx 9.35\%$ MAE reduction.

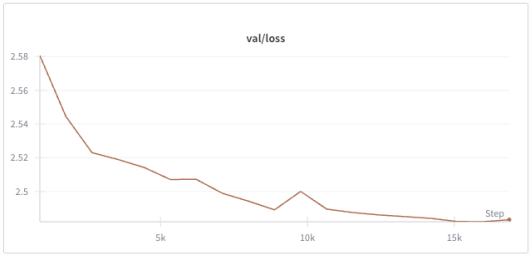
How performance scales with history



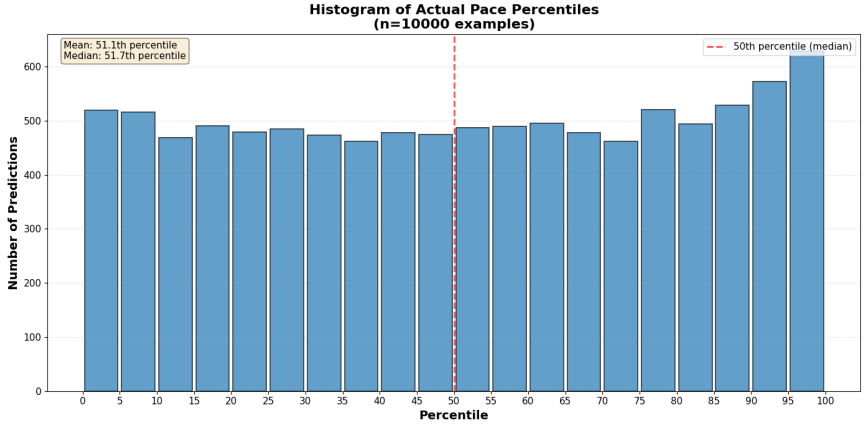
Training curves (W&B exports)



Validation curves (W&B exports)



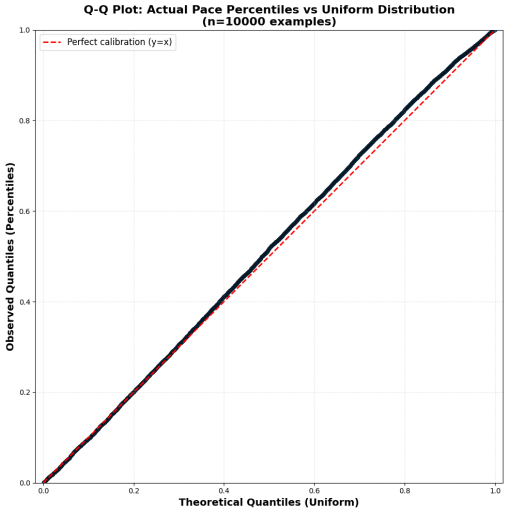
Percentile calibration histogram



Each bin spans 5 percentile points; n=10,000 predictions. Under perfect calibration every bin would hold 500 entries, and the near-uniform counts (34–69) show the PDF percentiles are well balanced before deeper diagnostics.

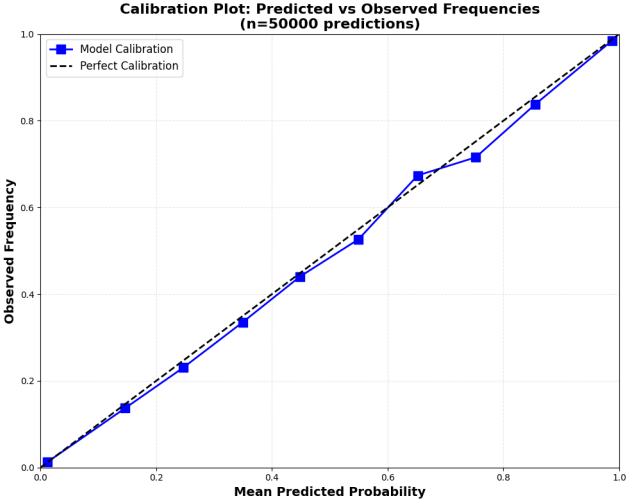
Quantile-quantile diagnostics

Q-Q plot: KS $D = 0.025$, $p < 0.001$; max deviation approximately 2.5pp. The straight diagonal alignment against Uniform(0,1) proves the distributional predictions stay balanced across bins. Deviations remain small even at the upper tail, explaining the low KS statistic.



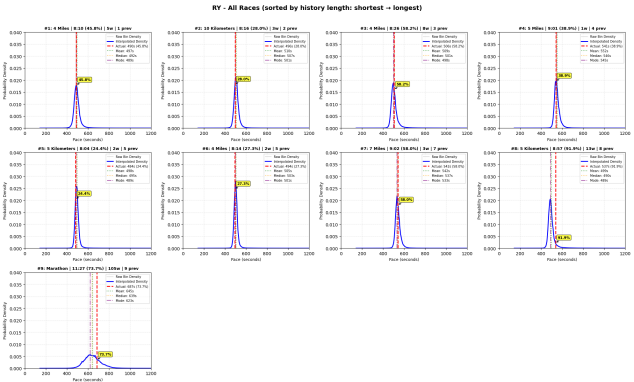
Calibration curve

Calibration curve: 50,000 threshold predictions keep deviations under 3 percentage points while the mean predicted (60.1%) matches the observed (59.7%). Every bin hugs the diagonal, so probability mass at each confidence level is trustworthy. This stability justifies acting on any quantile without systematic bias.



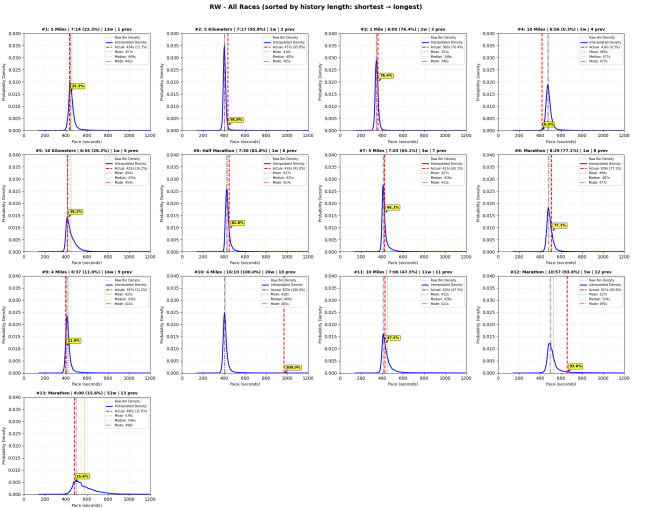
Interpretability: RY runner PDFs

RY (n=9) shows moderate uncertainty. Shorter races yield tight PDFs (45-58%ile), while the marathon (#9) becomes right-skewed with the actual outcome near 73.7%ile, capturing distance-sensitive volatility without explicit distance modeling. The distributions therefore confirm that context-specific uncertainty pathways arise naturally from the tokenized input.

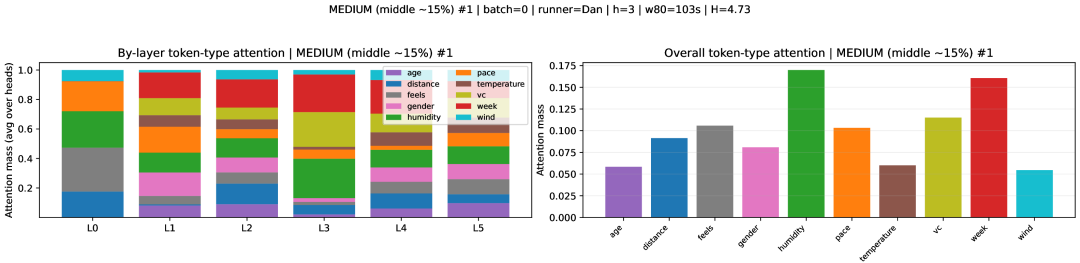


Interpretability: RW runner PDFs

RW (n=13) has the widest PDFs (0.015-0.025 peak), especially for marathons (16-100%ile). Wide, asymmetric distributions and extreme outcomes such as race #10 (100%ile) and #13 (15.6%ile) still sit in the tails, illustrating that the model quantifies the athlete's inherent variability instead of collapsing to overconfident point estimates.



Mechanistic inspection (example view)



By-layer token-type attention helps validate *what* the model uses (cadence vs context vs history) as a function of depth.

Generalization beyond this dataset

While motivated by irregular event sequences, the recipe applies broadly to regression and forecasting:

- define a tokenization (or block) for context
- choose an output distribution (discretized bins)
- train for distributional prediction (PDF) instead of point prediction

Not just irregular events:

- **h=0 (no history)**: predict from static covariates only (standard regression)
- **regular intervals**: apply on fixed grids (standard time series) and still output PDFs

This positions RunTime-style models as competitors to tabular baselines and classical forecasting (e.g., ARIMA), with the same distributional output interface.

Immediate follow ups (from the paper)

Follow-ups to isolate which components drive the gains and validate generality beyond NYRR:

- **Time-token ablation**: remove `d_next` / `d_fin` (or replace with a constant) to test performance without explicit gap knowledge
- **Swapped-token grammar**: move `pace` earlier to avoid artificial terminal delta tokens
- **Zero-history baseline (h=0)**: static covariates only under the same entity-disjoint split
- **External validation**: apply to MIMIC-IV to predict a time-to-next-admission PDF (patient-disjoint split)

Future application 1: system monitoring (health / maintenance)

Goal: move from static thresholds to individualized trajectory-aware risk.

- tokens encode irregular measurements + context (state, environment, interventions)
- output is a **PDF over next measurement** (or time-to-event via target inversion)
- enables **early warnings** when observed values fall in low-probability regions

Why PDF matters: decisions often depend on tail risk (rare but costly excursions).

Future application 2: engagement / churn

Goal: detect rhythm shifts before the terminal event (churn, dropout, relapse).

- events: sessions, purchases, support tickets, refills, check-ins
- cadence tokens capture **inter-arrival changes** and recency patterns
- output PDF can target outcome (e.g., next action) or time-to-next action

Why PDF matters: interventions can be triggered when probability mass shifts into risky tails.

Future application 3: hydrology & energy (extremes)

Goal: forecast distributions over extremes, not just averages.

- hydrology: PDFs over river height / levee-cross risk under irregular storm cadence
- energy: PDFs over localized peak demand / transformer stress during heat events
- evaluate by tail metrics (coverage at high quantiles), not only mean error

Why PDF matters: infrastructure cost is dominated by rare peaks and failures.

Future application 4: markets (tail-risk forecasting)

Goal: predict distributions over drawdowns, volatility spikes, and jumps.

- events: trades, regime changes, macro shocks; cadence can be irregular
- output PDF can target **drawdown size** or **time to next shock**
- supports stress testing and risk budgeting via quantiles/ES-like summaries

Why PDF matters: tail outcomes dominate hedging and capital requirements.

Practical note on data availability

To prevent abuse (e.g., automated scraping / bulk pulling of underlying raw results), **not all acquisition / raw-data retrieval steps are included** in the public repo.

- the repo remains runnable end-to-end on included sample shards
- interested researchers can reach out for additional details as appropriate

Repro & pointers

- Technical writeup: `Technical_Details.md`
- Baselines: `train/Benchmark_Baselines.py` (wrapper: `train/run_xgboost_tuning.sh`)
- Evaluation notebook: `train/Inspect_Model_Outputs.ipynb`
- Activation inspection: `train/Inspect_Model_Activations.ipynb`

Repo: `github.com/yaelelmatad/RunTime-Public`

Questions?

github.com/yaelelmatad/RunTime-Public