Timestamp Embeddings

John Ash

Independent researcher

First published 28 June 2026; revised 12 November 2026

Abstract

Sequence position identifies where an observation appears in a model's input. A timestamp identifies when it occurred. These coordinates serve different purposes, particularly when observations are irregularly spaced or sampled across a long history. This paper specifies the timestamp encoder used in the current Ŧrust systems. It maps one absolute Unix timestamp to a learned vector by combining a regularized trend pathway with a bank of learned periodic components through an adaptive gate. A two-part numerical transport preserves whole-second timestamps on float32 hardware while retaining the same absolute-time meaning. The representation supplies temporal structure to observations and queries, and composes with content and source embeddings inside ordinary transformer attention. The paper explains the implemented mechanism, its use in forecasting and source selection, and its integration into the Market Transformer and embedded trading policy. These working numerical systems provide a foundation for learning the temporal development of language, contributions, and patterns across sources.

1 Introduction

An observation made yesterday and an observation made ten years ago can occupy adjacent positions in a context window. Their adjacency says where the model encounters them, not how much time separates them. Likewise, a densely sampled week and a sparsely sampled decade can contain the same number of observations. Reasoning about these contexts requires information beyond their sequence indices.

A timestamp embedding gives the model an explicit representation of when each observation belongs in history. The input is one absolute time coordinate, expressed in Unix seconds. A learned function converts that coordinate into a vector that can participate in attention alongside the observation's content and source.

The representation combines two forms of temporal structure. A trend pathway can encode nonperiodic change. A periodic pathway can encode recurring relationships at learned scales. An adaptive gate combines their outputs. This supplies the model with temporal features through which it can learn, for example, that a seasonal observation is relevant to a later season or that a source's usefulness changes through time.

The current implementation is used in the Market Transformer and its embedded decision policy. Earlier numerical experiments demonstrate its underlying design in forecasting and in learning which source to read at a given time. This paper defines the current absolute-time implementation, distinguishes its numerical transport from the semantic input, and explains how it supports the joint source–time–content architecture described in the Ŧrust white paper [1].

2 Position, time, and related work

Positional encodings supply the ordering information used by transformer attention [2]. They remain useful when the order of words or other elements within a sequence matters. Timestamps supply a different coordinate: the actual time associated with an observation. A language model can use token positions within a passage and a timestamp for the passage itself. One does not replace the other.

Written dates and temporal descriptions can also supply information through language. An explicit timestamp encoder makes the numerical coordinate available directly, with a consistent meaning across date formats and documents. It lets a model learn temporal relationships through the encoding's operations instead of requiring every relationship to be expressed in surrounding prose.

Learned representations of time have an established literature. Time2Vec combines a linear component with learned periodic components in a representation that can be used by different architectures [3]. Xu et al. develop functional time representations using translation-invariant kernels and learned feature maps [4]. T-Rep learns time embeddings alongside a feature extractor for self-supervised time-series representation learning [5]. Temporal Attention for Language Models incorporates document time into transformer self-attention and studies the resulting representations through semantic change [6].

The encoder described here combines a regularized nonlinear trend branch, a periodic bank initialized across multiple frequency bands, and adaptive routing between the branches. Its current implementation retains absolute time and uses compensated floating-point arithmetic for phase calculations. These are the specific design choices developed below; the cited methods establish the broader setting for learning temporal representations.

3 The current timestamp encoder

3.1 One semantic input

The encoder receives an absolute Unix timestamp τ: seconds since the Unix epoch. It does not receive a second feature measured in days, a calendar decomposition, or a position relative to a particular dataset. The output is a vector $e_t(\tau) \in \mathbb{R}^d$.

The current data path holds whole-second timestamps as CPU `int64` values. The encoder's mathematical form is a learned function of a real-valued time coordinate; the implemented transport preserves the whole-second inputs used by these systems. A different timestamp resolution would require a corresponding input representation.

Within one fixed model state, the same absolute timestamp has the same representation wherever it is used. Training changes the learned parameters, and dropout is active during training. In evaluation mode, the encoding is deterministic and does not depend on dataset boundaries.

3.2 Trend pathway

The trend branch supplies a learned representation of nonperiodic change. Internally, the timestamp is divided by the fixed constant 86,400 before entering a small multilayer perceptron. This changes the numerical scale of the same input; it does not add an independently supplied days feature.

Let LN denote layer normalization, GELU the activation function, and $\mathrm{Drop}_{0.5}$ dropout with probability 0.5 during training. The branch is

$$u(\tau) = 0.1\ \mathrm{LN}_u\big(W_2\ \mathrm{GELU}\big[\mathrm{Drop}_{0.5}(W_1\,\tau/86400 + b_1)\big] + b_2\big)\,. \tag{1}$$

Here W_1 maps a scalar to d dimensions and W_2 maps d dimensions to d dimensions. The first weight matrix is initialized uniformly in $[-10^{-5}, 10^{-5}]$. The small initialization, dropout, and final factor of 0.1 regularize and scale the trend contribution while the model learns how to use it.

3.3 Periodic pathway

The periodic branch supplies a bank of learnable oscillators. Each dimension has a log-frequency parameter f_j and phase ϕ_j; a shared learned scale α multiplies the frequencies. The branch is

$$\omega_j = \exp(f_j)\alpha, \qquad v(\tau) = \mathrm{LN}_v\left(\left[\sin(\tau\omega_j + \phi_j)\right]_{j=1}^{d}\right). \tag{2}$$

The implementation initializes equal numbers of log-frequencies in four bands: $[-20, -10]$, $[-10, 0]$, $[0, 6]$, and $[6, 12]$. The dimension d is chosen to be divisible by four. Phases are initialized from a standard normal distribution, and α starts at one.

This initialization supplies candidate periodic features across a broad range of scales. Training adjusts their frequencies and phases according to the objective. The model can learn relevant periodicities without receiving separate hour, weekday, month, or season labels. The bank remains a learned set of oscillators; selection of useful information occurs through the trained encoder and downstream computation.

3.4 Adaptive gate

The two branches produce vectors of the same width. A gate receives their concatenation and outputs two nonnegative weights that sum to one. Writing sg for stop-gradient, the implemented routing and output are

$$g(\tau) = \text{softmax}(G_2\ \text{GELU}[G_1\ \text{sg}([u(\tau); v(\tau)]) + a_1] + a_2)\,, \tag{3}$$

$$e_t(\tau) = g_1(\tau)u(\tau) + g_2(\tau)v(\tau). \tag{4}$$

The gate's hidden width is $4d$, and its first linear layer uses spectral normalization. The gate is shared across timestamps and can vary its allocation according to the branch representations at a particular time.

The stop-gradient operation has a precise role. It blocks the indirect gradient route from the gate's input back into the branch representations. Both branches still receive gradients through their weighted contributions in Equation 4, and the gate's own parameters are trained. Spectral normalization controls the first gate layer's weight scale. These operations specify how routing is learned; they do not impose a fixed allocation between trend and periodic structure.

4 Preserving absolute time on float32 hardware

4.1 Exact transport of the timestamp

Modern Unix timestamps are large numbers. Converting one directly to float32 can make distinct nearby seconds indistinguishable before the periodic calculation begins. This matters because the phase in Equation 2 depends on the timestamp multiplied by a learned frequency.

The current `TimestampEncodingV11` implementation transports the timestamp as two float32 components. On the CPU, the whole-second input is converted through float64 to form

$$\tau_{\text{hi}} = \text{float32}(\tau), \qquad \tau_{\text{lo}} = \text{float32}\left(\tau - \text{float64}(\tau_{\text{hi}})\right). \tag{5}$$

The split helper checks that adding these components in float64 reconstructs the original integer second. Both components are measured in seconds. Together they transport one timestamp; they are not separate temporal features, and they do not change the architecture's semantic input.

The model can then move the pair to MPS or another float32 device without collapsing it into one rounded scalar. The trend branch computes its internal rescaling as

$$\tau/86400 \;\longrightarrow\; \tau_{\text{hi}}/86400 + \tau_{\text{lo}}/86400. \tag{6}$$

4.2 Compensated periodic phase calculation

For the periodic branch, the implementation multiplies each component by the learned frequencies using compensated products. It combines the high and residual terms with compensated sums, performs range reduction using a two-part representation of 2π, and then adds the learned phase before applying the sine function.

This preserves low-order timestamp information through the large phase calculation. The input reconstruction is exact for the checked whole-second timestamps; the subsequent products, range reduction, and sine

remain floating-point numerical calculations. The distinction is between exact transport of the input coordinate and compensated evaluation of the learned function.

The current encoder has no dataset-minimum, dataset-maximum, or dataset-range buffers. Earlier experiments included a normalized variant that expressed time relative to a dataset's bounds. The current version uses absolute seconds throughout, with the fixed internal trend rescaling described above. Learned parameters specify the clock; a dataset's chosen start and end do not redefine it.

5 Integration with attention

5.1 Observations and requests share a clock

A temporal model needs to know both when its evidence was observed and when an answer is requested. The same encoder supplies those coordinates. In forecasting, each context observation combines a content vector with the encoding of its observation time. A request for a value at a target time uses the encoding of that target as part of its query.

This lets attention compare the request with context through learned temporal features. A model can learn to use recurring relationships, nonperiodic change, or both. The encoding supplies the features; the task objective determines which temporal relationships matter to the answer.

Timestamp spacing is independent of the number of context entries. A fixed number of observations can be sampled densely around a recent event and sparsely across a longer history. Their absolute times remain available to the model. The context budget determines how many observations are supplied; the timestamps determine where those observations lie in time.

The treatment of ordering still belongs to the surrounding architecture. Unmasked attention over a set of paired observations can preserve its result under joint permutation of those observations. The current market model uses chronologically arranged context and a causal mask, so its attention structure also carries ordering constraints. Timestamp embeddings operate within those constraints rather than removing them.

5.2 Composition with source identity

The current numerical systems combine content, time, and source in four compatible parts. Let c_i denote embedded content, $p_i = P_t e_t(\tau_i)$ the clock output projected to the context width, and z_{s_i} the learned source representation. The observation becomes

$$x_i = [c_i; p_i; z_{s_i}; p_i \odot z_{s_i}]. \tag{7}$$

The source representation is a registered embedding lookup followed by a learned shared projection [7]. The final term is an elementwise source–time interaction. It makes the contribution of a particular source at a particular time directly available to attention.

In the current Market Transformer and embedded policy, the clock produces a vector of width 192, which is projected to width 48 for this construction. Four parts of width 48 form the contextual representation. The content part combines numerical value and return encodings. A source-and-time query uses the corresponding time, source, and interaction terms with a zero content slot for the requested unknown.

The Market Transformer develops this representation through contextual layers, a query-addressed evidence read, and further learned computation. Its numerical output heads produce trajectory information. The embedded policy retains the representation before those output heads and uses it in its own decision computation. The timestamp module therefore contributes to both numerical generation and practical action [1].

Information availability is enforced by the surrounding data and attention paths. In particular, an issued prediction may refer to a future moment while already being available as evidence. The time represented by an observation and the time at which it became available have to be respected by the task's context

construction. A timestamp embedding represents its supplied coordinate; it does not independently decide which evidence the model is allowed to use.

6 Numerical demonstrations

6.1 Forecasting from irregular context

The forecasting experiment asks whether explicit timestamps help predict a value from observations taken at irregular intervals. Its synthetic signal combines a directional trend, periodic components, and noise. Context values are projected into vectors and combined with their timestamp representations. Self-attention processes the context, and a target-time query reads from it to predict the requested value.

In the random-context condition, the sampler includes randomly chosen earlier observations and the observation immediately preceding the target. The times therefore reveal both the irregular gaps and the location of the nearby observation. Training and evaluation are separated by target time. The comparison includes the original timestamp encoder, the earlier normalized variant, and a model without timestamp encoding.

The recorded run shows that the encoded models learn the forecasting task substantially better than the model without time. The relevant result is that the model can use the actual observation and query times to recover temporal structure from irregular context. This experiment supports the dual-pathway design; the later compensated transport preserves absolute timestamps for its use in the current systems.

6.2 Learning source reliability through time

The source-selection experiment changes the question from which value occurs at a time to whose offered value should be used at that time. Several sources contribute numerical predictions. A periodic rule determines which source is currently the expert. That source supplies the target value, while the others supply noise from the same marginal distribution.

The model receives source-labelled predictions at multiple timestamps and learns from the numerical answer alone. No expert-source label enters the training loss. The full content–time–source representation learns to select the relevant contribution. Reductions that remove source, time, or both remain near chance on this task. Removing time also replaces timestamp-query attention with pooling, so these are comparisons of the available representations and readout paths.

The sample windows span different temporal ranges and are divided chronologically by their first timestamp. Separate inspections examine which offered value the model's answer selects and which source its attention emphasizes. Together, these establish a working use of time and source identity for learning changing reliability. The historical proof uses the earlier normalized clock variant; the current market and policy implementation uses the absolute-time encoder specified in Sections 3 and 4.

6.3 Current system use

The mechanism continues into the Market Transformer and the embedded trading policy. Both use the absolute-time encoder on source-labelled numerical observations. The embedded policy reads the resulting latent field together with portfolio and recurrent state, learning decisions about buying, selling, holding, and trade size [1].

The current policy loader reconstructs the checkpoint-associated model and restores the policy's saved parameters. Its clock remains `TimestampEncodingV11`, with the same trend branch, periodic bank, detached gate input, and compensated timestamp transport described here. End-to-end training can update that clock; serving evaluates the saved model. This is the current operational use of the component, beyond its earlier controlled demonstrations.

7 Next steps

The next step is to bring the same temporal coordinate into a text model trained on attributed, timestamped contributions. Token positions would preserve local linguistic order, while timestamps would connect observations across documents and historical intervals. The model could then learn how a concept, claim, or source's contribution develops through time rather than seeing only isolated textual snapshots.

The broader purpose is to learn from track records and the evolution of a shared body of knowledge. A contribution can be novel before it is widely accepted. Time makes it possible to relate that contribution to later evidence, subsequent refinements, and changing use. Training on those relationships requires outcome and evidence signals that distinguish warranted revision from repetition or popularity.

Source-indexed attention offers a corresponding route to studying credit: which contributors introduced, developed, tested, or combined a pattern, and which of their contributions the model uses for a request. Extending the current numerical systems to language and media requires preserving those relationships through the model and evaluating their role in its outputs. The timestamp encoder supplies the temporal coordinate for that work.

8 Conclusion

Timestamp embeddings make absolute time available as a learned input to attention. The current encoder combines nonperiodic trend and learned periodic structure through an adaptive gate, while its two-part transport preserves the whole-second coordinate on float32 hardware. The same clock can represent observation times and requested times without redefining time around each dataset.

The numerical systems use this mechanism for forecasting, source-dependent reasoning, market representations, and practical decisions. Combined with source and content, it lets the model learn relationships across contributors and moments within the same attention architecture. That working foundation supports the next stage: learning the development of language and collective contributions through time, with their temporal and source relationships retained.

References

[1] J. Ash. Ŧrust: Learning where patterns come from and how they change through time. 2026. https://www.speakerjohnash.com/literature/trust-white-paper.pdf

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017. https://arxiv.org/abs/1706.03762

[3] S. M. Kazemi, R. Goel, S. Eghbali, J. Ramanan, J. Sahota, S. Thakur, et al. Time2Vec: Learning a vector representation of time. 2019. https://arxiv.org/abs/1907.05321

[4] D. Xu, C. Ruan, S. Kumar, E. Korpeoglu, and K. Achan. Self-attention with functional time representation learning. *Advances in Neural Information Processing Systems*, 32, 2019. https://arxiv.org/abs/1911.12864

[5] A. Fraikin, A. Bennetot, and S. Allassonnière. T-Rep: Representation learning for time series using time-embeddings. *International conference on learning representations*, 2024. https://arxiv.org/abs/2310.04486

[6] G. D. Rosin and K. Radinsky. Temporal attention for language models. *Findings of the association for computational linguistics: NAACL 2022*, pp. 1498–1508, 2022. https://aclanthology.org/2022.findings-naacl.112/

[7] J. Ash. Source embeddings. 2026. https://www.speakerjohnash.com/literature/source-embeddings.pdf