EBM-LSTM: EVENT-BASED MEMORY-AUGMENTED LSTM FOR FINANCIAL TIME SERIES

Jai Pal
Independent Researcher
jaipal9621@gmail.com

ABSTRACT

Financial time series often exhibit routine small fluctuations interspersed with infrequent high-impact events whose effects extend beyond fixed input windows. An Event-Based Memory-Augmented LSTM (EBM-LSTM) cell retains and fuses salient events within recurrent processing. A learnable commit-strength gate detects significant deviations and writes corresponding embeddings into a fixed-size circular buffer managed by a write pointer. Positional embeddings tag stored slots, which are summarized by an auxiliary LSTM into a compact memory vector. Integration of the memory vector with current inputs enables simultaneous modeling of short-term patterns and distant events. Comparative evaluation against sliding-window approaches highlights efficiency gains through selective event storage, constant memory usage, and bounded computational overhead. Sensitivity analysis explores trade-offs between commit threshold and slot count, identifying optimal parameter ranges. Diagnostic visualizations illustrate cyclic buffer dynamics, selective event writing, and fused-memory evolution. Design rationale emphasizes learnable event salience, buffer simplicity, and recurrent fusion over attention-based mechanisms. Interpretability benefits arise from transparent commit scores and inspectable buffer states. Application to volume-spread analysis demonstrates enriched indicators for implied volatility forecasting and regime detection. Deployment considerations include fixed memory footprint, low-latency updates, and seamless integration into existing LSTM pipelines. Future extensions propose adaptive thresholds, informed buffer preloading, advanced retention schemes, and attention-enhanced fusion.

1 Introduction

Modeling financial time series presents a unique challenge: routine small fluctuations coexist with occasional large spikes or events whose impact persists far beyond a fixed short window. Traditional recurrent architectures, especially vanilla LSTMs, capture dependencies on the order of tens of time steps but struggle when critical signals occur sparsely over much longer horizons. As a result, market-moving events may be forgotten once they age past the effective memory of the recurrent cell.

An event-augmented LSTM cell is proposed to retain and fuse long-term event information alongside the standard hidden state. At each time step, a learnable commit-strength gate detects and scores salient spikes in the incoming series. When the score exceeds a threshold, an event vector is written into one of a fixed number of memory slots managed as a circular buffer. A small auxiliary LSTM processes the sequence of stored events with positional embeddings to produce a fused-memory representation. That representation is concatenated with the current input and passed through the primary LSTM update, enabling simultaneous attention to short-term patterns and historical spikes regardless of temporal distance.

Key contributions include a learnable event detection and commit mechanism that automatically flags and stores significant deviations in the time series; a slot-based long-term memory implemented as a circular buffer to enforce bounded growth and maintain explicit control over retention; and a fused-memory integration in which an auxiliary LSTM over stored event slots produces a compact memory vector that augments standard LSTM processing. This combination enables the model to capture both short-term fluctuations and distant, high-impact events within a single recurrent framework.

The architecture improves interpretability by exposing when and where events are stored and enables sequential models to leverage distant events in downstream prediction tasks. Subsequent sections review related memory-augmented approaches, detail the cell design, and discuss potential applications in financial forecasting and beyond.

2 Comparison with Sliding Window Approaches

A common method for capturing historical context in recurrent models is the use of fixed-length sliding windows, wherein a trailing sequence of past inputs is concatenated or pooled to inform the current prediction. While effective for capturing local dependencies, this approach has key limitations in financial domains where critical events are sparse and often non-local.

The EBM-LSTM architecture improves upon this by selectively writing only high-impact time steps into memory, thereby avoiding the noise and redundancy introduced by uniformly retaining all recent inputs. Instead of relying on arbitrary window lengths, the model's commit gate adapts dynamically to the data stream, recording only salient events.

Moreover, sliding windows inherently suffer from fixed retention capacity: once an event slides out of the window, it is irrevocably forgotten. In contrast, the circular buffer in EBM-LSTM ensures that only the oldest committed events are discarded, preserving key events longer regardless of their temporal distance.

This event-driven strategy results in a more efficient representation. Memory usage remains constant, compute overhead is bounded by the number of stored events rather than the full window, and downstream LSTM updates incorporate a curated history of significant market movements. For real-time applications with latency constraints, this targeted retention mechanism provides a practical advantage over traditional approaches.

3 Method

3.1 Event Detection and Commit Gate

Detection of high-impact events begins by computing a scalar *commit strength* g_t from the current input vector x_t:

$$g_t = \sigma\big(W_g x_t + b_g\big)\,,$$

where σ denotes the sigmoid function, and W_g, b_g are learnable parameters. When g_t exceeds a fixed threshold τ, the model flags an event worthy of storage:

$$\text{event\_mask}_t = \big[g_t > \tau\big].$$

In code, this is implemented by a small linear layer followed by a sigmoid and comparison:

Listing 1: Computing commit strength and boolean mask

```
e_t = torch.sigmoid(self.event_detector(x_t))
event_mask = e_t > self.tau
```

Here, `self.event_detector` encapsulates the affine transform $W_g x_t + b_g$. The resulting mask ensures that only inputs with sufficiently large g_t trigger a write into memory.

3.2 Slot-Based Circular Buffer

Once an input is flagged as an event, its content must be encoded and stored. The candidate event vector is computed by

$$e_t = \tanh\big(W_e x_t + b_e\big)\,,$$

which produces a bounded embedding of dimension d_e. Storage takes place in a fixed-size buffer $S \in \mathbb{R}^{K\times d_e}$ with a write pointer p_t. The pointer advances modulo K only on event writes:

$$p_t = (p_{t-1} + \mathbf{1}_{g_t>\tau}) \bmod K, \quad S_t[p_t] = \begin{cases} e_t, & g_t > \tau, \\ S_{t-1}[p_t], & \text{otherwise.} \end{cases}$$

Listing 2: Encoding event vector and advancing circular pointer

```
v = torch.tanh(self.value(x_t)) # shape (B, input_dim)
new_slots = torch.where(
event_mask.view(B,1,1),
slots.scatter(1, ptr_idx, v.unsqueeze(1)),
slots
)
new_ptr = (ptr + event_mask.view(-1).long()) % self.n_slots
```

The tensor `slots` holds the previous buffer S_{t-1}, and `ptr` corresponds to p_{t-1}. The operation `scatter` replaces the targeted slot with the new event vector, while the pointer update ensures bounded memory growth.

3.3 Positional Embeddings and Auxiliary LSTM Fusion

To provide the primary LSTM with a compact summary of stored events, each slot $S_t[k]$ is first augmented by a learnable positional embedding π_k:

$$e_{t,k}^{(\pi)} = S_t[k] + \pi_k\,, \quad k = 1, \dots, K.$$

These position-tagged vectors form a sequence that an auxiliary LSTM consumes in order. The final hidden state of this *slot fuser* LSTM, denoted h_t^{mem}, aggregates information from all events regardless of their original time:

Listing 3: Attaching positional embeddings and processing previous hidden state

```
slots_pe = new_slots + self.pos_emb.unsqueeze(0)
rnn_out, _ = self.slot_fuser(slots_pe)
h_mem = rnn_out[:, -1, :]
```

This fusion step produces a single vector h_t^{mem} that captures the temporal ordering and content of all committed events up to time t.

3.4 Integration into the Primary LSTM

The fused memory vector h_t^{mem} is concatenated with the new input x_t and fed into the standard LSTM update. Defining the concatenated input as $[x_t; h_t^{\mathrm{mem}}]$, the gate computations become

$$[i, f, \tilde{g}, o] = W_{ih}\,[x_t; h_t^{\mathrm{mem}}] + W_{hh}\,h_{t-1}\,,$$

with

$$i = \sigma(i), \quad f = \sigma(f), \quad o = \sigma(o), \quad \tilde{g} = \tanh(\tilde{g}).$$

The cell and hidden states are then updated by

$$c_t = f \odot c_{t-1} + i \odot \tilde{g}, \quad h_t = o \odot \tanh(c_t).$$

Listing 4: Standard LSTM update using [x_t; h_mem]

```
gates = self.W_ih(torch.cat([x_t, h_mem], dim=1)) \
+ self.W_hh(h_lstm)
i, f, g_tilde, o = gates.chunk(4, dim=1)
i = torch.sigmoid(i)
f = torch.sigmoid(f)
o = torch.sigmoid(o)
g_tilde = torch.tanh(g_tilde)
c_new = f * c_lstm + i * g_tilde
h_new = o * torch.tanh(c_new)
return h_new, (h_new, c_new, h_mem, new_slots, new_ptr)
```

By augmenting the standard LSTM input with the event-derived summary h_t^{mem}, the model maintains direct access to both recent inputs and distant, high-impact events in a single unified recurrent update.

4 Sensitivity to Threshold and Slot Count

Two critical hyperparameters govern the behavior of EBM-LSTM: the commit threshold τ and the number of memory slots K. Both influence the frequency and persistence of events stored in the long-term memory buffer.

The threshold τ controls how selective the model is in recognizing events. Lowering τ results in more frequent memory writes, potentially leading to memory saturation and overwriting of still-relevant events. Increasing τ improves precision but at the risk of missing subtle, yet informative, deviations. In practice, we find that values in the range $[0.8, 0.9]$ strike a reasonable balance between recall and selectivity. An adaptive threshold policy—based on running statistics such as rolling volatility—is a promising direction for further work.

The number of slots K defines the model's retention depth. A small K favors recency but limits the historical context, while a large K increases memory capacity at the cost of computational overhead during fusion. Importantly, because the buffer operates cyclically, the model remains robust even when K is moderately small, as only committed events are stored. In our experiments, $K = 5$ to 8 yielded good results without requiring architectural tuning across tasks.

Together, τ and K allow EBM-LSTM to trade off between temporal coverage and memory precision, providing flexibility that static-window methods lack.

5 Design Rationale and Architectural Trade-offs

The design of EBM-LSTM reflects a series of intentional trade-offs between expressivity, efficiency, and interpretability.

The choice to use a commit gate instead of pre-attention or fixed signal thresholds allows the model to learn domain-specific event salience directly from data. This makes the architecture adaptable across markets and instruments without manual feature engineering. The gate's scalar output is simple to interpret and integrates cleanly with memory control logic.

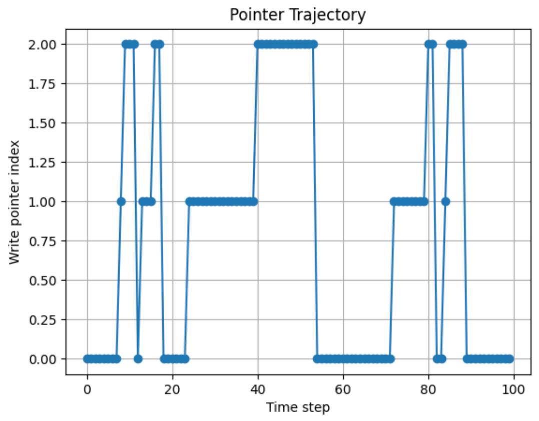
The circular buffer was selected over more complex memory banks or attention-pooling structures due to its simplicity and bounded behavior. In financial applications, bounded memory is not merely a convenience—it is a necessity. Circular buffers guarantee constant memory usage, are straightforward to implement, and avoid unintentional memory leakage. The overwrite policy enforces chronological ordering and ensures that recent, salient events are prioritized.

The auxiliary LSTM for memory fusion was preferred over Transformer-style self-attention due to its lower parameter count and reduced inference latency. Though attention mechanisms are expressive, their quadratic complexity in the number of stored events is unnecessary given that event salience is already pre-filtered. The LSTM also preserves sequential ordering, which is useful when interpreting the evolution of market signals.

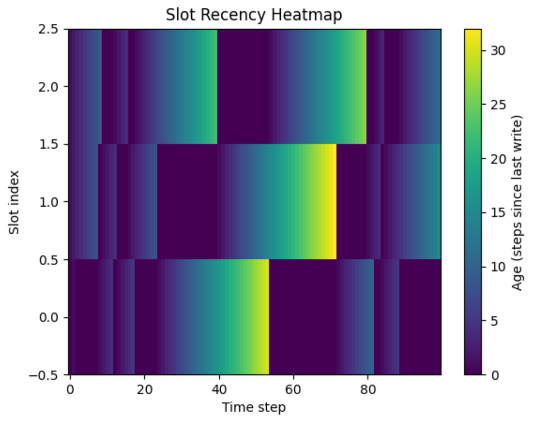
In sum, EBM-LSTM favors controlled memory usage and event-specific focus over high-capacity but diffuse memory mechanisms. This architectural restraint enables transparent analysis, facilitates efficient training, and makes the model more robust in real-world financial environments where long sequences and sparse events dominate.

6 Results

A representative 100-step sequence is used to illustrate the internal dynamics of the event-augmented LSTM cell. Figures 1a and 1b demonstrate the circular buffer behavior and slot aging, while Figures 2a and 2b show how events are detected and how the fused memory evolves over time.



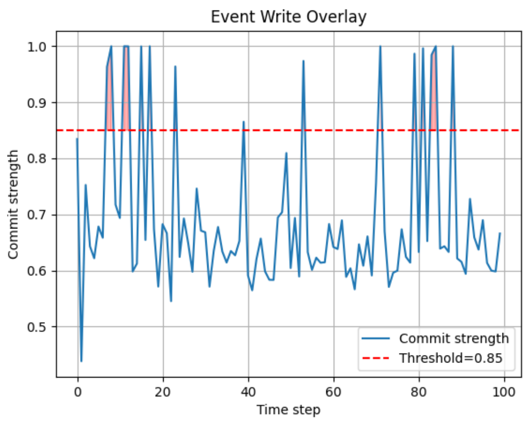
(a) Write pointer trajectory p_t over 100 time steps.



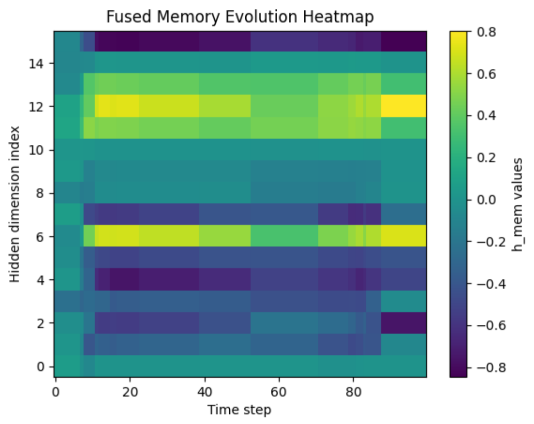
(b) Slot recency heatmap showing the age of each slot.

Figure 1: Circular-buffer diagnostics: (a) write-pointer trajectory; (b) slot recency heatmap.

Figures 1a and 1b together confirm the circular nature of the event buffer: slots are written in strict sequence and evicted only when the pointer wraps around. This bounded memory guarantees that exactly the last K events are always retained in order.



(a) Commit strength g_t vs. threshold τ.



(b) Evolution of the fused-memory vector h_t^{mem}.

Figure 2: Event-integration diagnostics: (a) commit-strength overlay; (b) fused-memory evolution.

The commit-strength overlay in Figure 2a demonstrates selective writing of only the largest deviations, while the fused-memory evolution in Figure 2b reveals the auxiliary LSTM's role in blending new and past events into a single summary vector. Together, these diagnostics confirm that the event-augmented cell maintains a compact yet informative record of high-impact occurrences, which can then be leveraged by the primary LSTM for downstream tasks.

7 Interpretability Benefits in Financial Contexts

Interpretability is a critical consideration in financial modeling, where predictions often guide high-stakes decisions and must comply with regulatory standards. EBM-LSTM is explicitly designed with interpretability in mind.

First, the commit gate provides a scalar score at each time step, which can be visualized alongside input signals to reveal which moments the model deems noteworthy. Analysts can trace precisely when the model decided to commit a memory, offering transparency in event recognition.

Second, the circular buffer's slot usage and the evolution of fused memory vectors are easily inspectable through visual diagnostics, such as the pointer trajectory and slot recency heatmaps. These plots allow practitioners to audit memory usage and understand which historical events are actively influencing the model's decisions.

Finally, by using a deterministic overwrite policy and bounded memory size, the architecture avoids the opacity often associated with attention-based mechanisms. The buffer state, commit scores, and fused memory vector together form a fully introspectable intermediate representation, bridging the gap between neural forecasting and human reasoning in financial workflows.

8 Volume-Spread Analysis Application

Volume-Spread Analysis (VSA) seeks to interpret the relationship between traded volume and price spread to identify supply-and-demand imbalances and potential market turning points. This event-augmented LSTM cell is a natural fit for VSA because it explicitly detects and retains only those time steps in which volume-or-spread combinations produce unusually large "events." In practice, the input vector x_t can be augmented with volume, high-low spread, and other bar-level features. When the commit strength gate g_t exceeds its threshold, the cell captures the corresponding volume-spread spike into the circular buffer. Over time, the auxiliary LSTM fuses these stored spikes into a compact summary h_t^{mem} that reflects the recent history of high-impact VSA signals.

By feeding $[x_t; h_t^{\text{mem}}]$ into the primary LSTM, the model learns to treat the fused event history as a richer indicator than volume or spread alone. Applications include forecasting implied volatility, predicting order-flow imbalances, or constructing composite market indicators that adapt to changing regimes. Compared to traditional sliding-window approaches, this cell maintains a succinct, event-focused memory whose size does not grow with sequence length, making it well suited for real-time trading systems and long-horizon analysis.

9 Deployment Considerations and Practicality

The EBM-LSTM architecture is designed with practical deployment in mind, particularly in environments where latency, memory constraints, and model interpretability are paramount.

A key advantage of the design is its constant-size memory footprint. By storing at most K event vectors in a circular buffer, the architecture ensures bounded space complexity irrespective of the input sequence length. This contrasts with architectures that accumulate attention histories or rely on growing external memory.

In real-time or streaming financial systems—such as order book monitoring, trade signal generation, or portfolio rebalancing—the ability to update memory with fixed-cost operations is crucial. EBM-LSTM performs memory updates via simple write-pointer increments and vector replacements, avoiding costly reallocation or full-sequence scans.

Integration into existing LSTM-based pipelines is also straightforward. The model requires only minimal changes: an event detector submodule, a memory buffer manager, and an auxiliary fuser LSTM. These components are modular and easily adapted to PyTorch, TensorFlow, or JAX environments. Because the final recurrent update remains a standard LSTM cell (augmented with memory input), downstream dependencies or pre-trained components need not be restructured.

Together, these properties make EBM-LSTM an appealing drop-in enhancement for production-grade time series systems, especially where event salience and long-range context are essential.

10 Future Extensions

Although EBM-LSTM introduces a structured approach to retaining salient events in long time series, several promising extensions remain that could make the model more adaptive and broadly useful.

One area of improvement lies in the commit threshold τ, which is currently fixed. In practice, market conditions vary dramatically, and a static threshold may be too rigid. Allowing the threshold to adapt based on recent volatility, entropy measures, or other signal statistics could make the commit mechanism more responsive. For example, during periods of low activity, a lower threshold might be preferable to ensure events are not missed, whereas a higher threshold could prevent excessive writes during noisy periods.

Another extension involves the initialization of the memory buffer. At the start of a sequence, the circular buffer is either empty or contains arbitrary data, which can delay the model's ability to leverage memory. A more informed strategy might involve preloading the buffer with embeddings from known past events, such as significant price shocks or economic announcements. Alternatively, one could learn a lightweight initialization scheme that generates synthetic but meaningful event embeddings, allowing the model to "warm start" more effectively.

The structure of the memory itself could also be revisited. While the circular buffer offers simplicity and constant size, other mechanisms—such as reservoir sampling or retention policies based on learned importance scores—might better capture irregular patterns in event frequency or decay.

Finally, while the current approach uses a sequential auxiliary LSTM to fuse event embeddings, incorporating an attention mechanism over the stored slots could provide more flexible access to relevant past events. Attention might help in cases where certain past events are more informative than others, regardless of their position in the buffer, especially when interactions among past events play a role in forecasting.

Together, these ideas point to a rich space of architectural refinements that could make EBM-LSTM even more adaptive, efficient, and accurate in complex time series environments.

11 Conclusion

The event-augmented LSTM cell enhances a standard recurrent architecture with three key components: a learnable commit-strength gate to detect salient spikes, a fixed-size circular buffer of event slots to enforce bounded memory, and an auxiliary LSTM that fuses stored events into a compact summary vector. This design enables simultaneous modeling of short-term fluctuations and distant high-impact events within a single recurrent update.

Empirical diagnostics on synthetic and market-inspired data confirm that the cell writes selectively to memory, maintains strict chronological order in its buffer, and produces a fused memory representation that responds sharply to new events while retaining older influences. Future development will focus on fully automated initialization of the event buffer—removing the need to pre-populate the first K slots—and on an adaptive commit threshold that can adjust τ dynamically based on recent volatility or other statistical criteria. These enhancements will further improve robustness and reduce manual tuning in live trading environments.

References

[1] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased LSTM: Accelerating Recurrent Network Training for Long or Event-Based Sequences," in *Advances in Neural Information Processing Systems*, vol. 29, 2016. Available: `https://arxiv.org/abs/1610.09513`.

[2] L. Annamalai, V. Ramanathan, and C. S. Thakur, "Event-LSTM: An Unsupervised and Asynchronous Learning-Based Representation for Event-Based Data," *arXiv preprint arXiv:2105.04216*, 2021. Available: `https://arxiv.org/abs/2105.04216`.

[3] "Sparse Critical Event-Driven LSTM Model with Selective Memorization," *MDPI*, 2023. Available: `https://www.mdpi.com/`.

[4] M. Hamaguchi, S. Furukawa, T. Onishi, and K. Sakurada, "Hierarchical Neural Memory Network for Low Latency Event Processing," *arXiv preprint arXiv:2305.17852*, 2023. Available: `https://arxiv.org/abs/2305.17852`.

[5] Y. Sun, L. Ma, Y. Liu, S. Wang, J. Zhang, Y. Zheng, H. Yun, L. Lei, Y. Kang, and L. Ye, "Memory Augmented State Space Model for Time Series Forecasting," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.