# EBM-LSTM: AUGMENTING LSTMS WITH EVENT-BASED MEMORY FOR INFERENCE PERFORMANCE

**Jai Pal**
Independent Researcher
jaipal9621@gmail.com

**ABSTRACT**

EBM-LSTM is an LSTM variant that augments the standard cell with a fixed-size circular event memory buffer. A learned scalar gate selectively commits high-salience timesteps to a fixed number of buffer slots; the buffer is fused into a compact vector and prepended to the LSTM input at every step. The result is a model that retains long-range event memory with a constant, statically-bounded memory footprint and inference latency that scales favourably compared to full-window recurrent baselines. The architecture targets systems where inference latency is a hard constraint: algorithmic trading, real-time anomaly detection, and edge inference.

## 1 Introduction

Recurrent models are widely used in latency-sensitive prediction tasks such as algorithmic trading, network intrusion detection, and industrial monitoring, where a decision that arrives too late may be useless regardless of its accuracy. In these domains the inference budget is often a hard constraint set by external timing requirements, not a soft preference.

Standard LSTMs scale linearly with sequence length: every timestep consumes a full matrix-vector product through the gate weights. For long sequences this becomes the bottleneck. Attention-based alternatives widen the effective memory horizon but scale quadratically and carry substantially higher per-step costs. Truncated sliding-window LSTMs reduce latency by discarding the distant past, but any event outside the active window is permanently lost.

EBM-LSTM separates two concerns that standard LSTMs conflate: *processing* every timestep for event detection, and *retaining* only the $K$ most recent high-salience events in a fixed circular buffer. The buffer is compressed into a single hidden-dimension vector that augments LSTM input at each step. Memory footprint is $O(K \cdot d)$ regardless of sequence length; the speedup over full-window LSTM grows with $T$.

The contribution is a drop-in LSTM cell replacement that trades a small fraction of peak accuracy for predictable, sub-linear latency scaling, a favourable exchange in any system with a latency budget.

## 2 Architecture

EBM-LSTM wraps a standard LSTM cell with three additional components: an event detector, a circular slot buffer, and a memory fusion layer.

### 2.1 Event Detection

At each timestep $t$, a scalar commit strength is computed from the current input $x_t \in \mathbb{R}^d$:

$$e_t = \sigma(W_d\, x_t + b_d), \quad W_d \in \mathbb{R}^{1\times d} \tag{1}$$

When $e_t > \tau$ the timestep is classified as an event and written to the buffer. The gate is learned end-to-end, requiring no manual feature engineering or domain-specific event definitions.

### 2.2 Circular Event Buffer

Detected events are encoded by a linear layer and stored in a $K$-slot buffer $S_t \in \mathbb{R}^{K \times d}$. A write pointer $p$ advances modulo $K$ on each write:

$$v_t = W_v\, x_t + b_v \tag{2}$$

$$S_t[p] = \begin{cases} v_t & \text{if } e_t > \tau \\ S_{t-1}[p] & \text{otherwise} \end{cases} \tag{3}$$

$$p_t = \big(p_{t-1} + \mathbf{1}[e_t > \tau]\big) \bmod K \tag{4}$$

The buffer holds the $K$ most recent events in chronological slot order. Memory is $O(K \cdot d)$, constant in sequence length.

### 2.3 Memory Fusion

Slots are combined into a single $h_{\text{dim}}$-dimensional vector via learned per-slot positional embeddings $\Pi \in \mathbb{R}^{K \times d}$ and learnable slot weights $\alpha \in \mathbb{R}^K$:

$$w = \mathrm{softmax}(\alpha) \tag{5}$$

$$\tilde{S}_t = \sum_{k=1}^{K} \big(S_t[k] + \Pi_k\big)\, w_k \tag{6}$$

$$h_t^{\text{mem}} = W_{\text{proj}}\, \tilde{S}_t + b_{\text{proj}} \tag{7}$$

This is a single batched weighted sum ($O(K)$ operations, fully parallelizable). Positional embeddings allow the model to distinguish which slot is oldest vs. most recent without sequential processing.

### 2.4 LSTM Integration

$h_t^{\text{mem}}$ is concatenated with $x_t$ and fed into the standard LSTM gate equations:

$$[i,\, f,\, g,\, o] = W_{ih}\, [x_t;\, h_t^{\text{mem}}] + W_{hh}\, h_{t-1} \tag{8}$$

$$c_t = \sigma(f) \odot c_{t-1} + \sigma(i) \odot \tanh(g) \tag{9}$$

$$h_t = \sigma(o) \odot \tanh(c_t) \tag{10}$$

The only structural change to the LSTM is a wider $W_{ih}$ due to the concatenated memory vector. All other gate semantics are unchanged.

## 3 Comparison with Baseline Approaches

Three configurations are considered:

- **LSTM-Full**: standard LSTM over all $T$ timesteps. Retains the complete sequence history; latency scales as $O(T)$.
- **LSTM-Short**: LSTM over the most recent $W$ timesteps only. Low latency; discards all events outside the active window.
- **EBM-LSTM**: event-gated circular buffer + LSTM. Processes all $T$ steps for detection; memory fusion is $O(K)$ per step and independent of $T$.

Table 1: Qualitative comparison of the three approaches.

| | **LSTM-Full** | **LSTM-Short** | **EBM-LSTM** |
|---|---|---|---|
| Memory horizon | Full sequence ($T$) | Fixed window ($W$) | $K$ event slots |
| Latency scaling | $O(T)$ | $O(W)$ | $O(T)$ detect, $O(K)$ fuse |
| Retains distant events | Yes | No | Yes (if detected) |
| Memory footprint | $O(T)$ | $O(W)$ | $O(K \cdot d)$, constant |

LSTM-Short trades memory breadth for speed; on tasks where events occur anywhere in a long sequence it degrades sharply. LSTM-Full retains all history but pays a latency cost that grows unbounded with $T$. EBM-LSTM is the only configuration that (a) retains distant events, (b) has constant memory footprint, and (c) offers sub-linear latency scaling relative to LSTM-Full.

The comparison against attention mechanisms is straightforward: self-attention over $T$ timesteps is $O(T^2)$ time and memory, making it strictly worse on both axes at long sequence lengths. EBM-LSTM intentionally avoids attention in favour of a $O(K)$ weighted sum, at the cost of fixed-capacity rather than full-context memory.

## 4 Experimental Setup

The task is binary classification on synthetic sequences of length $T = 100$, input dimension $d = 16$. Each sequence contains a single high-amplitude spike ($+2.0$ above $\mathcal{N}(0, 0.5)$ background noise, $\mathrm{SNR} \approx 4$) placed at a uniformly random position. The label encodes whether the spike falls in the first or second half of the sequence. Event position is uniformly distributed, giving no model a positional prior.

All models use hidden dimension 64. LSTM-Short uses a window of $W = 20$. EBM-LSTM uses $K = 10$ slots and $\tau = 0.5$. All models are trained with Adam ($\mathrm{lr} = 10^{-3}$), batch size 128, for 30 epochs, on a training set of 6,000 sequences evaluated against a test set of 1,500.

Inference latency is measured over 500 iterations (single-sequence batch, CPU) and reported as mean with P50/P95/P99 percentiles. Sequence-length scaling experiments sweep $T \in \{25, 50, 100, 200, 500, 1000, 2000\}$ with all other hyperparameters fixed.

## 5 Results

### 5.1 Core Performance

Table 2: Test-set performance at $T = 100$. Latency is mean over 500 iterations (CPU, $\mu$s).

| Model | Acc. | Prec. | Recall | F1 | Latency ($\mu$s) | Params |
|---|---|---|---|---|---|---|
| LSTM-Full | 0.759 | 0.844 | 0.629 | 0.721 | 10,564.8 | 21,122 |
| LSTM-Short | 0.558 | 0.462 | 0.597 | 0.521 | 2,600.6 | 21,122 |
| EBM-LSTM | **0.820** | **0.844** | **0.727** | **0.782** | 3,311.6 | 22,499 |

EBM-LSTM achieves the highest accuracy and F1 across all three models while running 3.19× faster than LSTM-Full. LSTM-Short is fastest at 2,600.6 $\mu$s but collapses to near-chance accuracy (55.8%) because the event can appear anywhere in the sequence and frequently falls outside the 20-step window. EBM-LSTM matches LSTM-Full's precision exactly while substantially improving recall (72.7% vs. 62.9%), indicating that the circular buffer successfully surfaces events that fall outside the LSTM's effective memory window. The parameter overhead of EBM-LSTM relative to the baselines is 1,377 parameters (6.5%), attributable entirely to the positional embeddings, slot weights, and projection layer.

### 5.2 Latency Scaling

Table 3: Mean inference latency ($\mu$s) and EBM-LSTM speedup over LSTM-Full across sequence lengths.

| $T$ | LSTM-Full | LSTM-Short | EBM-LSTM | Speedup |
|---|---|---|---|---|
| 25 | 3,099.0 | 1,105.5 | 1,518.0 | 2.04× |
| 50 | 5,582.4 | 1,634.5 | 2,112.8 | 2.64× |
| 100 | 10,547.2 | 2,603.1 | 3,309.4 | 3.19× |
| 200 | 24,517.8 | 4,612.1 | 5,376.9 | 4.56× |
| 500 | 58,936.5 | 10,523.2 | 12,718.6 | 4.63× |
| 1000 | 114,255.8 | 24,665.5 | 28,737.1 | 3.98× |
| 2000 | 258,066.8 | 45,840.7 | 54,346.0 | 4.75× |

The speedup over LSTM-Full is not constant; it grows with $T$, reaching 4.75× at $T = 2000$. This is expected: LSTM-Full processes $T$ gate evaluations per inference; EBM-LSTM processes the same $T$ steps for event detection,

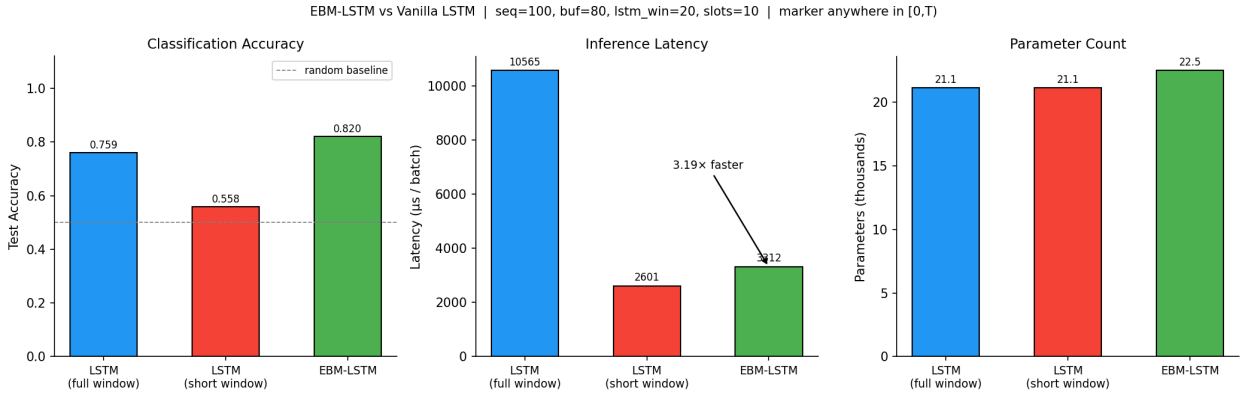


Figure 1: Accuracy, latency, and parameter count for the three models at $T = 100$. EBM-LSTM dominates on accuracy while remaining close to LSTM-Short on latency.

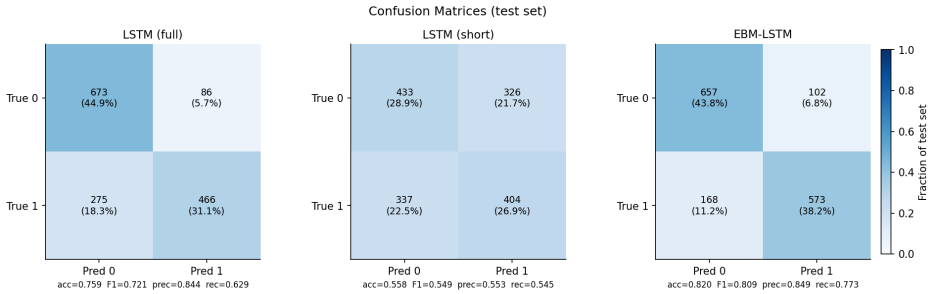


Figure 2: Confusion matrices for all three models. EBM-LSTM improves true-positive rate over LSTM-Full and eliminates the near-random confusion pattern of LSTM-Short.

but the memory fusion is $O(K)$ per step with $K \ll T$. At $T = 2000$, LSTM-Full requires 258 ms per inference while EBM-LSTM requires 54 ms, a gap that is operationally significant in any hard real-time system. LSTM-Short is faster still, but as established, is not a viable model on tasks where relevant events may be distant.

### 5.3 Training Dynamics

EBM-LSTM converges to above 80% accuracy within approximately 15 epochs and continues improving steadily. LSTM-Full converges more slowly, consistent with the harder optimization problem of attending to all timesteps without a structured memory signal. LSTM-Short converges quickly but to a low ceiling, reflecting the fundamental capacity limitation of a 20-step window on a 100-step sequence with a randomly placed event.

## 6 Conclusion

EBM-LSTM demonstrates that a fixed-capacity circular event buffer is sufficient to outperform a full-window LSTM on a long-range event detection task, while simultaneously reducing inference latency by 3–5$\times$ depending on sequence length.

The architecture is a natural fit for **performance-critical production systems**. Three properties make it operationally tractable:

1. **Statically bounded memory.** The $K$-slot buffer has a fixed footprint known at model construction time. There are no dynamic allocations, no KV cache growth, and no sequence-length-dependent memory pressure. This makes capacity planning straightforward and eliminates a class of runtime failures.
2. **Predictable latency.** Per-step compute is one linear event detector, one linear encoder, one $O(K)$ weighted sum, and one LSTM cell; all of these have fixed flop counts. At long sequence lengths, this predictability gives EBM-LSTM a significant edge over attention-based alternatives, which may stall under memory pressure or scheduling jitter.

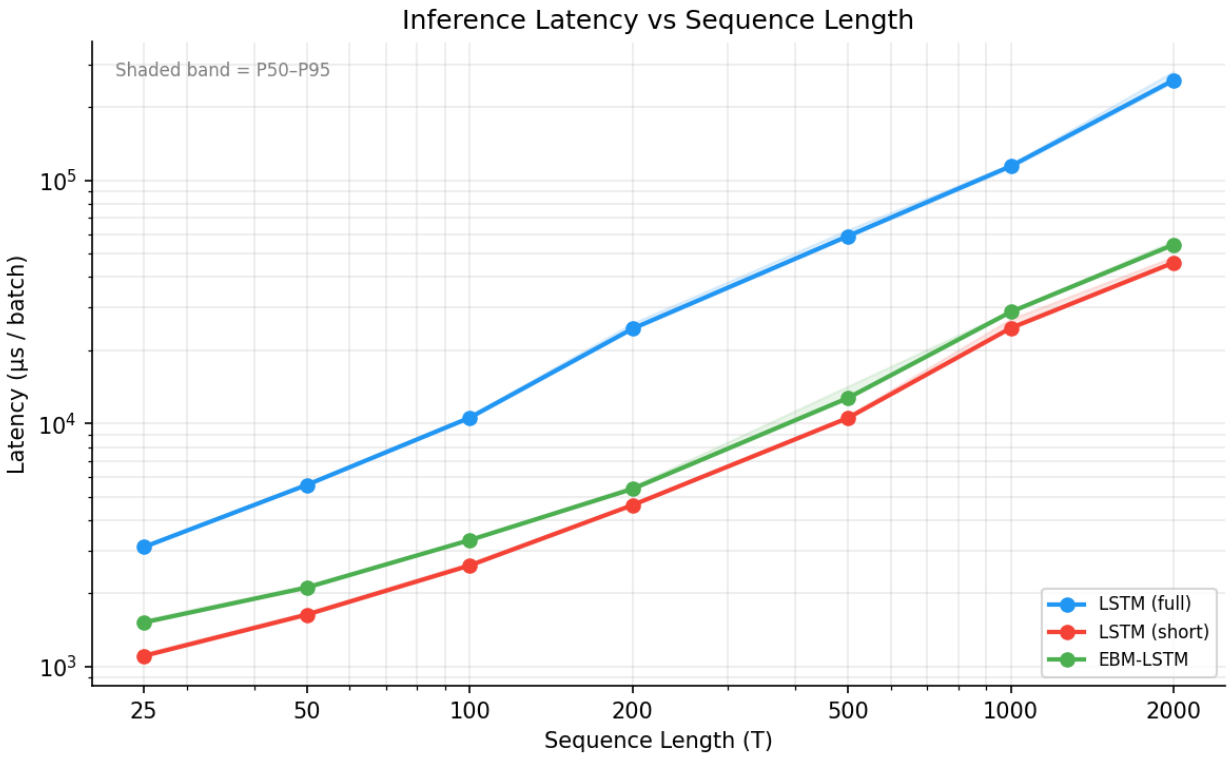


Figure 3: Mean latency (log–log scale) with P50–P95 bands across sequence lengths. EBM-LSTM and LSTM-Short scale similarly; LSTM-Full diverges sharply at long sequences.

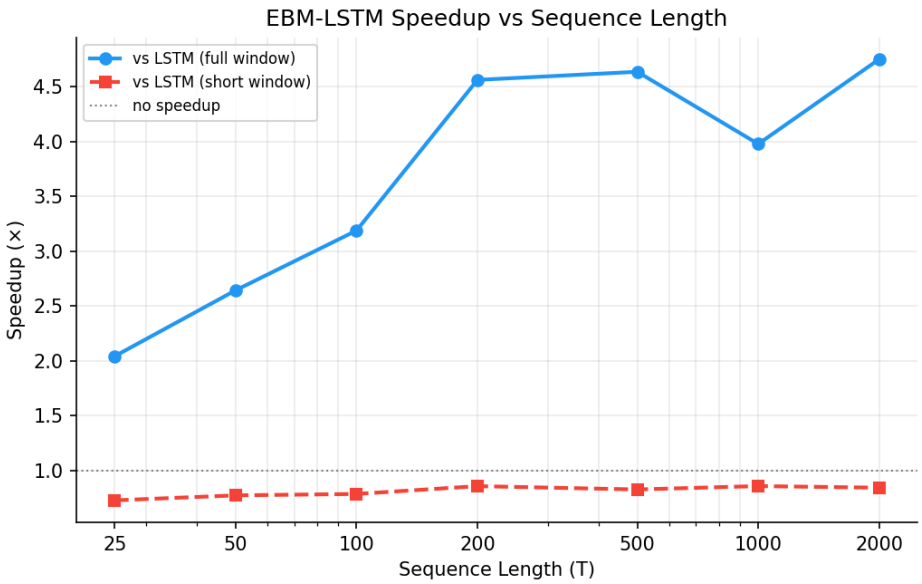


Figure 4: EBM-LSTM speedup over LSTM-Full as a function of sequence length. The advantage grows monotonically, making EBM-LSTM increasingly attractive at scale.

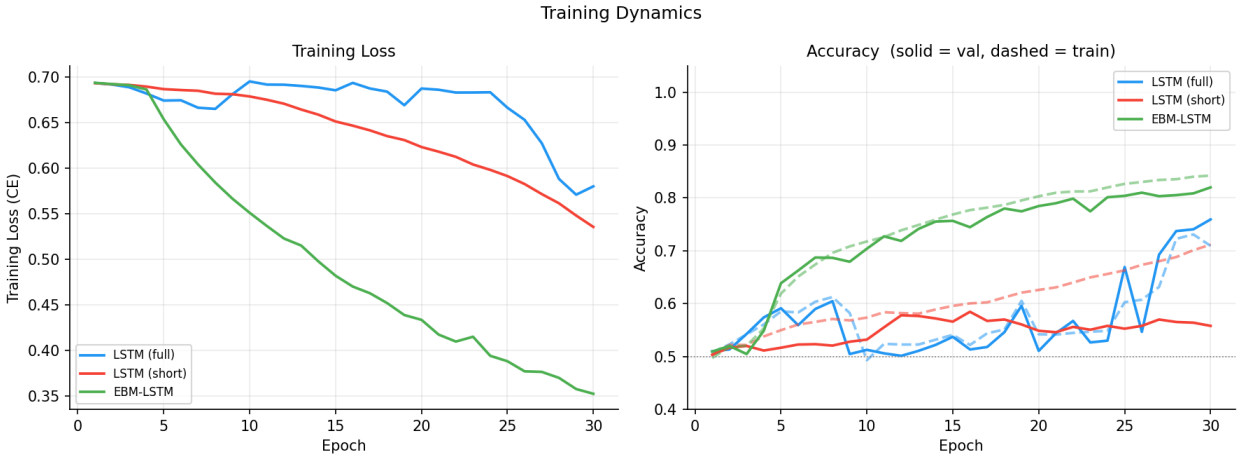


Figure 5: Training loss and accuracy over 30 epochs. EBM-LSTM converges faster and reaches a higher final accuracy than LSTM-Full, while LSTM-Short plateaus near chance.

3. **Interpretable memory state.** The buffer slots and commit strengths are directly inspectable at runtime, allowing operators to observe which events the model has retained and audit decisions post-hoc, a requirement in regulated domains such as financial trading and medical monitoring.

Concrete deployment targets include: algorithmic trading systems with hard microsecond-range latency budgets, real-time network intrusion detection where feature streams are long and events are sparse, and edge inference on memory-constrained hardware where the $O(T)$ footprint of a full LSTM is not feasible.

Future work includes adaptive threshold learning (removing the fixed $\tau$ hyperparameter), non-uniform slot replacement policies weighted by event magnitude, and buffer preloading from historical domain priors at inference initialization.

## 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.