Supplemental Material: Streaming Semi-CRF Inference

This supplement provides formal algorithmic details for the streaming Semi-CRF implementation.

1 Notation

Symbol	Description	Shape
T	Padded sequence length (batch dimension)	scalar
L_b	True (unpadded) length for batch element b	scalar
K	Maximum segment duration	scalar
C	Number of labels (states)	scalar
C_{PAD}	Padded label count (next power of 2)	scalar
B	Batch size	scalar
$\mathcal{S}_{t,c}$	Cumulative centered scores	$(B, T+1, C)$
$f_\theta(t, c)$	Raw projected encoder emissions	(B, T, C)
$\nu_{b,c}$	Sequence-level emission baseline (Eq. 1)	(B, C)
$\bar{f}_\theta(t, c)$	Centered emissions: $f_\theta(t, c) - \nu_{b,c}$	(B, T, C)
$\mathcal{T}_{c',c}$	Transition scores (source c' to dest c)	(C, C)
$\mathcal{B}_{k,c}$	Duration bias for duration k, label c	(K, C)
$\mathcal{P}^{\text{start}}_{t,c}$	Start boundary projection (optional)	(B, T, C)
$\mathcal{P}^{\text{end}}_{t,c}$	End boundary projection (optional)	(B, T, C)
$\tilde{\alpha}_t(c)$	Log-forward message at position t, label c	(B, C)
$\tilde{\beta}_t(c)$	Log-backward message	(B, C)
$\boldsymbol{\alpha}$	Forward ring buffer	(B, K, C)
$\boldsymbol{\beta}$	Backward ring buffer	$(B, 2K, C)$
Ω	Checkpointed ring buffer states	(B, N, K, C)
$\mathcal{N}$	Cumulative log-normalization factors	(B, N)
Δ	Checkpoint interval ($\approx \sqrt{TK}$)	scalar
τ	Tile size for label dimension	scalar
$\log Z$	Log partition function	$(B,)$

Table 1: Notation. Tilde ($\tilde{\cdot}$) denotes log-domain quantities. $N = \lceil T/\Delta \rceil$ is the number of checkpoints.

Indexing conventions. Positions $t \in \{0, \ldots, T\}$ index *boundaries* between tokens; cumulative scores $\mathcal{S}_{t,c}$ are defined at boundaries, with $\mathcal{S}_{0,c} = 0$. Emissions $f_\theta(t, c)$ are defined at tokens $t \in \{0, \ldots, T-1\}$ (0-indexed). Segments are half-open intervals $[s, s+k)$ covering tokens $s, s+1, \ldots, s+k-1$, so that the content score is $\mathcal{S}_{s+k,c} - \mathcal{S}_{s,c}$. Durations are 1-indexed: $k \in \{1, \ldots, K\}$, stored at 0-indexed positions $k-1$ in code.

2 Edge Potential Decomposition

The key innovation enabling $O(KC)$ memory is computing edge potentials on-the-fly from cumulative scores. We first describe the construction of these cumulative scores, then the edge potential formula.

2.1 Emission Baseline Centering

Let $f_\theta(t, c)$ denote the raw projected encoder emissions at token t for label c. We introduce *globally-centered emissions*, a formulation that couples local segment scores to global label statistics via a

per-label, per-sequence baseline subtraction. While primarily motivated by numerical stability at $T > 10^5$, this formulation induces an implicit adaptive duration prior (Section 9.1) that we find beneficial for segmentation stability in label-imbalanced sequences.

Before constructing the prefix sums, we subtract the *sequence-level emission baseline*:

$$\nu_{b,c} = \frac{1}{T} \sum_{u=0}^{T-1} f_\theta(u, c), \qquad \bar{f}_\theta(t, c) = f_\theta(t, c) - \nu_{b,c} \tag{1}$$

where T is the padded sequence length.[1] Globally-centered emissions are part of the model definition; all reported results use this formulation unless otherwise stated.
For each batch element b, the cumulative scores are then defined over the centered emissions:

$$\mathcal{S}_{0,c} = 0, \qquad \mathcal{S}_{t,c} = \sum_{u=0}^{t-1} \bar{f}_\theta(u, c) \tag{2}$$

so that the content score for a segment $[s, s + k)$ labeled c becomes

$$\mathcal{S}_{s+k,c} - \mathcal{S}_{s,c} = \sum_{u=s}^{s+k-1} f_\theta(u, c) \; - \; \nu_{b,c} \cdot k \tag{3}$$

Equation (3) shows that centering introduces a data-dependent term $-\nu_{b,c} \cdot k$ that couples label identity to segment duration. Equivalently, the effective duration model for label c is $\mathcal{B}_{k,c} - \nu_{b,c} \cdot k$, where the second term is an *adaptive duration prior* derived from the sequence-level emission statistics (Section 9.1).

2.2 Edge Potentials

The edge potential for a segment ending at position t with duration k, transitioning from source label c' to destination label c, is:

$$\tilde{\psi}(t, k, c, c') = \underbrace{(\mathcal{S}_{t,c} - \mathcal{S}_{t-k,c})}_{\text{centered content}} + \underbrace{\mathcal{B}_{k,c}}_{\text{duration bias}} + \underbrace{\mathcal{T}_{c',c}}_{\text{transition}} + \underbrace{\mathcal{P}^{\text{start}}_{t-k,c} + \mathcal{P}^{\text{end}}_{t-1,c}}_{\text{boundary (optional)}} \tag{4}$$

where $k \in \{1, \ldots, K\}$ is the duration, c is the destination state, and c' is the source state. The optional boundary projections $\mathcal{P}^{\text{start}}$ and $\mathcal{P}^{\text{end}}$ score segment start/end positions (e.g., for detecting boundary-specific features). The prefix-sum decomposition allows $O(1)$-in-k edge computation (constant-time in the duration) instead of $O(k)$ accumulation.

3 Streaming Forward Algorithm

Algorithm 1 maintains a ring buffer $\boldsymbol{\alpha} \in \mathbb{R}^{K \times C}$ storing the K most recent forward messages (Figure 1a). Following the Flash Attention [1] and Mamba [2] pattern, we normalize at checkpoint boundaries to prevent scale drift and maintain numerical stability at extreme T.

Memory: $O(KC)$ ring buffer + $O(\sqrt{T/K} \cdot KC)$ checkpoints. **Time**: $O(TKC^2)$.

[1] The implementation computes the baseline over the full (padded) sequence via `scores_float.mean(dim=1, keepdim=True)`. Although tokens $u \geq L_b$ are masked in the DP (no segment may include them), they still influence $\nu_{b,c}$ and thus shift the centered emissions at all valid positions $u < L_b$. The effective model therefore depends on the padding emissions and the padded length T, not only on the true sequence. This definition is well-defined only relative to a fixed padding scheme (fixed T and fixed padding-token semantics). For the primary use case of fixed-length genomic windows, $L_b = T$ for all batch elements and the distinction vanishes. A masked-mean variant ($\nu_{b,c} = L_b^{-1} \sum_{u=0}^{L_b-1} f_\theta(u, c)$) would eliminate the padding dependence for heterogeneous batch lengths.

Algorithm 1: Streaming Semi-CRF Forward Scan

Input: $\mathcal{S}$, $\mathcal{T}$, $\mathcal{B}$, checkpoint interval Δ
Output: $\log Z$, checkpoints $(\Omega, \mathcal{N})$

$\boldsymbol{\alpha} \leftarrow -\infty$; $\quad \boldsymbol{\alpha}[0, :] \leftarrow 0$; $\quad \mathcal{N}_{\text{accum}} \leftarrow 0$;
$\Omega[0] \leftarrow \boldsymbol{\alpha}$; $\quad \mathcal{N}[0] \leftarrow 0$;
for $t \leftarrow 1$ **to** T **do**
  $\mathbf{v}_t \leftarrow -\infty \in \mathbb{R}^C$;
  for $k \leftarrow 1$ **to** $\min(K, t)$ **do**
    $\tilde{\boldsymbol{\alpha}}_{\text{prev}} \leftarrow \boldsymbol{\alpha}[(t-k) \bmod K, :]$;
    $\mathbf{h} \leftarrow \left(\mathcal{S}_{t,:} - \mathcal{S}_{t-k,:}\right) + \mathcal{B}_{k,:}$;
    $\mathbf{E} \leftarrow \mathbf{h}[:, \text{None}] + \mathcal{T}^\top$;
    $\mathbf{s}_k \leftarrow \text{LSE}(\tilde{\boldsymbol{\alpha}}_{\text{prev}}[\text{None}, :] + \mathbf{E}, \text{axis} = 1)$;
    $\mathbf{v}_t \leftarrow \text{LSE}(\mathbf{v}_t, \mathbf{s}_k)$;
  $\boldsymbol{\alpha}[t \bmod K, :] \leftarrow \mathbf{v}_t$;
  if $t \bmod \Delta = 0$ **then** `// Normalize at checkpoint [1]`
    $n \leftarrow t/\Delta$;
    `// Only normalize active sequences (` $t \le L_b$ `) to avoid phantom shifts`
    $s_b \leftarrow \begin{cases} \max_c \mathbf{v}_{t,b}(c) & \text{if } t \le L_b \\ 0 & \text{otherwise} \end{cases}$;
    $\mathcal{N}_{\text{accum}} \leftarrow \mathcal{N}_{\text{accum}} + s_b$; $\quad \boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} - s_b$;
    $\Omega[n] \leftarrow \boldsymbol{\alpha}$; $\quad \mathcal{N}[n] \leftarrow \mathcal{N}_{\text{accum}}$;
$\log Z \leftarrow \text{LSE}(\boldsymbol{\alpha}[T \bmod K, :]) + \mathcal{N}_{\text{accum}}$;
return $\log Z$, $(\Omega, \mathcal{N})$;

4 Backward Pass with Checkpointing

Algorithm 2 processes segments in reverse, recomputing α from checkpoints following gradient checkpointing [3] (Figure 1b). The saved $\mathcal{N}_i$ restores the true scale when computing marginals.

5 $2K$ Beta Ring Buffer

While the forward pass uses a ring buffer of size K (sufficient since we only look back K positions), the backward pass requires a ring buffer of size $2K$. This asymmetry arises from the different access patterns:

Forward pass access pattern. At position t, we read α values from positions $\{t-K, t-K+1, \ldots, t-1\}$—exactly K previous positions. A ring buffer indexed by $t \bmod K$ suffices.

Backward pass access pattern. At position t, we read β values from positions $\{t+1, t+2, \ldots, t+K\}$—$K$ future positions. However, we also write $\beta[t]$ while potentially still reading $\beta[t+K]$. If we used a ring buffer of size K, we would have:

$$\text{write index:} \quad t \bmod K \tag{5}$$

$$\text{read index for } k = K: \quad (t+K) \bmod K = t \bmod K \tag{6}$$

This creates a **write-before-read hazard**: we would overwrite $\beta[t+K]$ before reading it. The $2K$ ring buffer eliminates this conflict:

$$t \bmod 2K \neq (t+K) \bmod 2K \quad \text{for all } t \geq 0 \tag{7}$$

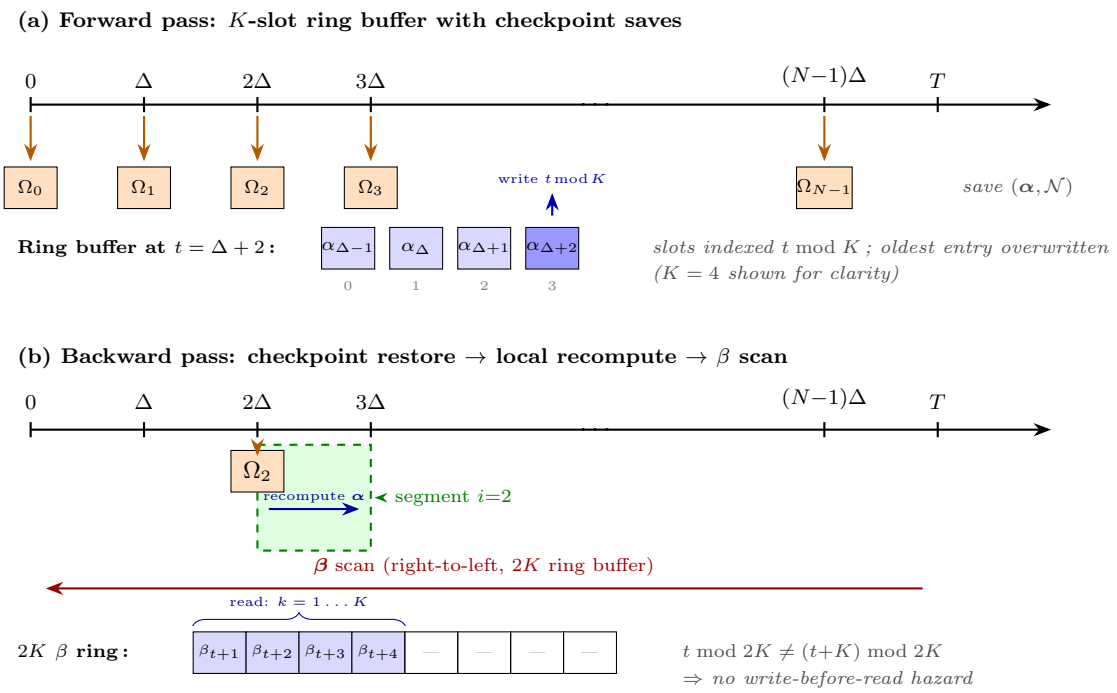

Figure 1: Memory layout for streaming semi-CRF inference. **(a)** During the forward pass, the ring buffer cycles through K slots via $t \bmod K$ indexing; at every Δ-th position, the full ring buffer state Ω_i and cumulative log-normalizer $\mathcal{N}_i$ are checkpointed. **(b)** During the backward pass, each checkpoint segment is processed independently: $\boldsymbol{\alpha}$ is restored from Ω_i and recomputed locally (blue arrow), while $\boldsymbol{\beta}$ is scanned right-to-left through a $2K$-slot ring buffer that prevents the write-before-read hazard (Section 5). Total memory is $O(KC)$ for ring buffers + $O(\sqrt{T/K} \cdot KC)$ for checkpoints, independent of sequence length T.

Memory cost. The additional KC floats is negligible compared to the $O(\sqrt{T/K} \cdot KC)$ checkpoint storage.

6 Adaptive Loop Tiling

The marginal computation requires a $(C_{\text{PAD}} \times C_{\text{PAD}})$ matrix per (t, k) pair, which at $C_{\text{PAD}} = 64$ demands approximately 384 registers per thread. With 4+ warps, this exceeds available registers and causes spilling to slow local memory.

Tiled computation. We process the destination label dimension c_{dst} in tiles of size τ (TILE_C):

1. Load a $(\tau \times C_{\text{PAD}})$ tile of the marginal matrix
2. Accumulate gradients from the tile
3. Use online logsumexp for β update across tiles (Flash Attention pattern [1])

This reduces peak register demand to approximately 120 per thread, enabling 4–8 warps without spilling.

Adaptive tile sizing. The tile size τ is selected based on C to balance compile time, register pressure, and iteration count:

Algorithm 2: Streaming Semi-CRF Backward Pass

Input: $\mathcal{S}$, $\mathcal{T}$, $\mathcal{B}$, checkpoints $(\Omega, \mathcal{N})$, $\log Z$, upstream $\partial\mathcal{L}/\partial Z$
Output: $\nabla\mathcal{S}$, $\nabla\mathcal{T}$, $\nabla\mathcal{B}$

$\boldsymbol{\beta} \leftarrow -\infty \in \mathbb{R}^{2K \times C}$; $\quad \boldsymbol{\beta}[T \bmod 2K, :] \leftarrow 0$;
Initialize gradient accumulators;
for $i \leftarrow N_{ckpts} - 1$ **to** 0 **do**
  $t_{\text{start}} \leftarrow i \cdot \Delta$; $\quad t_{\text{end}} \leftarrow \min((i+1) \cdot \Delta, T)$;
  $\mathcal{N}_i \leftarrow \mathcal{N}[i]$;
  $\boldsymbol{\alpha}_{\text{local}} \leftarrow$ RecomputeAlpha($\Omega[i]$, t_{start}, t_{end});
  for $t \leftarrow t_{end} - 1$ **to** t_{start} **do**
    $\tilde{\boldsymbol{\alpha}}_t \leftarrow \boldsymbol{\alpha}_{\text{local}}[t - t_{\text{start}}, :]$;
    $\tilde{\boldsymbol{\beta}}_t \leftarrow -\infty$;
    for $k \leftarrow 1$ **to** $\min(K, T - t)$ **do**
      $\tilde{\boldsymbol{\beta}}_{\text{next}} \leftarrow \boldsymbol{\beta}[(t+k) \bmod 2K, :]$; `// 2K ring buffer`
      $\tilde{\psi} \leftarrow$ Eq. (4);
      `// Log-norm correction restores true` α `scale`
      $\log \mu \leftarrow \tilde{\boldsymbol{\alpha}}_t[\text{None}, :] + \tilde{\psi} + \tilde{\boldsymbol{\beta}}_{\text{next}}[:, \text{None}] + \mathcal{N}_i - \log Z$;
      $\mu \leftarrow \exp(\log \mu)$;
      Accumulate $\nabla\mathcal{S}$, $\nabla\mathcal{T}$, $\nabla\mathcal{B}$ from μ;
      $\tilde{\boldsymbol{\beta}}_t \leftarrow \text{LSE}(\tilde{\boldsymbol{\beta}}_t, \text{LSE}(\tilde{\psi} + \tilde{\boldsymbol{\beta}}_{\text{next}}[:, \text{None}], \text{axis} = 0))$;
    $\boldsymbol{\beta}[t \bmod 2K, :] \leftarrow \tilde{\boldsymbol{\beta}}_t$; `// 2K ring buffer`
return $\nabla\mathcal{S}$, $\nabla\mathcal{T}$, $\nabla\mathcal{B}$;

Online logsumexp for β. Since β reduction spans multiple tiles, we use the Flash Attention online pattern:

$$m^{(i)} = \max(m^{(i-1)}, m^{(i)}_{\text{tile}}) \tag{8}$$

$$\ell^{(i)} = \ell^{(i-1)} \cdot e^{m^{(i-1)} - m^{(i)}} + \sum_j e^{x_j^{(i)} - m^{(i)}} \tag{9}$$

where m tracks the running maximum and ℓ tracks the running sum of exponentials, rescaled at each tile boundary.

7 Reduced Atomics Strategy

GPU atomic operations are expensive and introduce non-determinism due to floating-point non-associativity. We employ three strategies to minimize their use:

7.1 Local Accumulation for Per-Position Gradients

For $\nabla\mathcal{S}_{t,c}$, the negative contribution (from segments starting at t) is the same position for all k values. Instead of $K \times$ tiles atomic operations, we accumulate locally:

C_{PAD}	τ	Iterations	Rationale
≤ 8	4	2	Minimal iteration count
≤ 16	8	2	Minimal iteration count
32	16	2	Balanced
64	16	4	Moderate register pressure
≥ 128	32	≤ 8	Bounded compile time

Table 2: Adaptive tile size selection via `_compute_tile_c()`. The algorithm bounds iteration count to ≤ 8 even at $C = 256$ while keeping register pressure manageable.

Algorithm 3: Local Accumulation for $\nabla\mathcal{S}_t$

```
grad_cs_t_local ← 0 ∈ ℝ^{C_PAD};
for k ← 1 to K do
    for each tile do
        grad_cs_t_local −= Σ_{c'} μ_tile(t, k, c, c');        // Register accumulation
atomic_add(∇S_t, grad_cs_t_local);                            // Single write
```

Speedup: $K \times$ tiles $\to 1$ atomic per position (e.g., $1000 \times 4 = 4000 \to 1$ at $K = 1000$, $C = 64$).

7.2 Per-Duration Accumulation for Duration Bias

For $\nabla\mathcal{B}_{k,c}$, we accumulate across all tiles for each k, then write once:

Algorithm 4: Per-Duration Accumulation for $\nabla\mathcal{B}_k$

```
for k ← 1 to K do
    grad_db_k_local ← 0 ∈ ℝ^{C_PAD};
    for each tile do
        grad_db_k_local += Σ_{c'} μ_tile(t, k, c, c');
    atomic_add(∇B_k, grad_db_k_local);                        // Once per k
```

Speedup: tiles $\to 1$ atomic per (t, k) pair.

7.3 Segment-Isolated Workspace Buffers

For shared parameters ($\nabla\mathcal{T}$, $\nabla\mathcal{B}$), cross-segment atomic contention introduces non-determinism. We allocate per-segment workspace buffers:

$$\texttt{grad\_tr\_workspace} \in \mathbb{R}^{B \times N_{\text{segments}} \times C_{\text{PAD}} \times C_{\text{PAD}}} \tag{10}$$

(Padded to C_{PAD} to prevent out-of-bounds access from masked Triton threads; sliced back to C before reduction.)

Each checkpoint segment writes to its own slice, eliminating inter-segment atomics. The final reduction uses deterministic host-side operations:

$$\nabla\mathcal{T}_{c',c} = \texttt{einsum}(\text{“bsij, b} \to \text{ij”}, \texttt{workspace.sum(dim=1)}, \nabla_{\text{out}}) \tag{11}$$

$$\nabla\mathcal{B}_{k,c} = \texttt{einsum}(\text{“bskc, b} \to \text{kc”}, \texttt{workspace.sum(dim=1)}, \nabla_{\text{out}}) \tag{12}$$

Determinism: The segment-wise sum is deterministic (fixed order), and `einsum` performs a single reduction pass.

Memory cost: $O(B \cdot N_{\text{segments}} \cdot K \cdot C^2)$ workspace, which is acceptable since $N_{\text{segments}} \approx \sqrt{T/K}$ is typically small (e.g., 10–100 for $T = 100$k, $K = 1000$).

8 Gradient Computation

Gradients are computed via marginal probabilities:

$$\mu(t, k, c, c') = \exp\big(\tilde{\alpha}_{t-k}(c') + \tilde{\psi}(t, k, c, c') + \tilde{\beta}_t(c) + \mathcal{N}_i - \log Z\big) \tag{13}$$

For per-sequence parameters:

$$\nabla \mathcal{S}_{t,c} = \frac{\partial \mathcal{L}}{\partial Z_b} \cdot \left(\sum_{k,c'} \mu_b(t, k, c, c') - \sum_{k,c'} \mu_b(t + k, k, c, c') \right) \tag{14}$$

For shared parameters, we accumulate per-batch then reduce via einsum:

$$\nabla \mathcal{T}_{c',c} = \sum_b \left(\frac{\partial \mathcal{L}}{\partial Z_b} \cdot \sum_{t,k} \mu_b(t, k, c, c') \right) \tag{15}$$

$$\nabla \mathcal{B}_{k,c} = \sum_b \left(\frac{\partial \mathcal{L}}{\partial Z_b} \cdot \sum_{t,c'} \mu_b(t, k, c, c') \right) \tag{16}$$

The parentheses emphasize that each batch element's marginals are weighted by its upstream gradient *before* summing across batches. This distinction matters when $\partial \mathcal{L}/\partial Z_b$ varies (e.g., masked sequences or weighted losses).

9 Numerical Stability

9.1 Globally-Centered Emissions: Semantics and Motivation

Globally-centered emissions (Eq. 1) serve dual purposes: numerical stabilization of the prefix sums, and adaptive regularization of the duration model.

Numerical role. Without centering, cumulative scores grow as $\mathcal{S}_{T,c} = O(T)$ when encoder emissions have nonzero per-label means. At genomic scale ($T > 100{,}000$), segment scores $\mathcal{S}_{t+k,c} - \mathcal{S}_{t,c}$ risk catastrophic cancellation—both operands reach magnitude $O(T)$ while their difference is $O(1)$. Subtracting the per-label mean before accumulation reduces this to a zero-mean random walk whose cumulative sums have standard deviation $O(\sqrt{T})$ by the central limit theorem, analogous to how the log-sum-exp identity prevents overflow in softmax computation. The current implementation uses float64 for the forward-backward recurrence, which tolerates large input magnitudes; centering provides additional insurance, ensuring that a future migration to float32 kernels would not require algorithmic changes.

Modeling role. Global centering is *not* a path-invariant transformation: subtracting a label-specific constant $\nu_{b,c}$ shifts segment scores by $-\nu_{b,c} \cdot k$, which is label-dependent and length-dependent. A path-invariant centering (e.g., subtracting a per-position scalar $b_t = \max_c f_\theta(t, c)$ shared across labels) would cancel in the partition function because every valid segmentation covers each position exactly once. Per-label centering does not cancel because different segmentations assign different labels to different positions, and $\nu_{b,c}$ varies across labels.

Concretely, the effective segment score under centering (Eq. 3) is:

$$S_{\text{eff}}(c, s, k) = \sum_{u=s}^{s+k-1} f_\theta(u, c) + \mathcal{B}_{k,c} - \nu_{b,c} \cdot k \tag{17}$$

The term $-\nu_{b,c} \cdot k$ acts as a *data-dependent duration prior*: labels with high average emission $\nu_{b,c}$ incur a larger penalty for long segments, while labels with low average emission receive a relative boost.

This is equivalent to a canonical semi-CRF with an effective duration model $\mathcal{B}^{\text{eff}}_{k,c} = \mathcal{B}_{k,c} - \nu_{b,c} \cdot k$ that adapts per-sequence based on the global emission statistics.[2]

Implicit regularization: analogy to batch normalization. The dual character of emission centering—introduced for numerical stability, but inducing implicit regularization—parallels the role of batch normalization in deep networks [6]. Batch normalization was originally motivated by reducing internal covariate shift, but subsequent analysis showed that its primary benefit is smoothing the loss landscape [7]. Similarly, emission centering was introduced to bound prefix-sum magnitude, but its modeling side-effect—the adaptive duration prior—is arguably more consequential for segmentation quality.

The effective duration model $\mathcal{B}^{\text{eff}}_{k,c} = \mathcal{B}_{k,c} - \nu_{b,c} \cdot k$ has a natural interpretation as empirical Bayes shrinkage [8]: the learned parameters $\mathcal{B}_{k,c}$ capture population-level duration preferences across training sequences, while $\nu_{b,c}$ provides per-sequence adjustment based on the observed emission statistics. Labels whose emissions are unexpectedly high relative to the training distribution (large $\nu_{b,c}$) are shrunk toward shorter segments; labels with low emissions receive a relative boost. Since $\mathcal{B}_{k,c}$ can absorb the *average* centering effect across training, the adaptive prior primarily regularizes per-sample variation—sequences with atypical label composition receive atypical duration priors. The specific connection to segment durations in structured prediction, rather than to parameter estimates in the classical setting, appears to be novel.

In ablation experiments comparing per-label centering ("mean") against path-invariant centering ("position", which preserves the canonical semi-CRF distribution) on synthetic label-imbalanced sequences ($C = 4$, $T = 2{,}000$, proportions 70/15/10/5%), Viterbi decoding under mean centering recovered 24 segments of the rarest label versus 3 under path-invariant centering, and 43 versus 15 for the second-rarest label, while the dominant label's segment count decreased from 136 to 84. Under balanced label proportions (25% each), all centering modes produced nearly identical segmentations, confirming that the adaptive prior activates specifically in response to label imbalance rather than acting as a blanket regularizer.

Worked example: asymmetric regularization under label imbalance. Consider a sequence of $T = 10{,}000$ positions with $C = 3$ labels: Intron (dominant, 85% of positions), Exon (moderate, 14%), and Promoter (rare, 1%). Suppose the encoder produces well-calibrated emissions: each label emits high scores at its true positions and low scores elsewhere. Table 3 shows representative emission values, the resulting baselines $\nu_{b,c}$, and the adaptive duration penalty $-\nu_{b,c} \cdot k$ at segment length $k = 100$.

Three properties are visible from this example: (1) The duration penalty $-\nu_{b,c} \cdot k$ grows linearly in k, so it primarily affects *long* segments of the dominant label—exactly the degenerate segmentations one wishes to suppress. (2) Rare labels with low (or negative) baselines incur minimal penalty or even a slight boost, so their detection is not suppressed by centering. (3) Centered emissions at active positions ($f_\theta - \nu_{b,c}$) have *higher* contrast for rare labels (+8.12) than for dominant labels (+0.75), because centering subtracts a baseline that is close to the active-position emission for prevalent labels.

Architectural context. The impact of emission baseline centering depends on the CRF's role in the model architecture:

- *Encoder → CRF decoder.* When the CRF consumes encoder representations and produces a segmentation (e.g., as a structured output layer), the adaptive duration prior is consumed

[2] A semantics-preserving alternative exists: build prefix sums on the centered residuals $\bar{f}_\theta(t, c)$ but reconstruct the original segment score at evaluation time via $(\mathcal{S}_{s+k,c} - \mathcal{S}_{s,c}) + \nu_{b,c} \cdot k$. This retains the numerical benefit ($O(\sqrt{T})$ cumsum magnitude) without altering the semi-CRF distribution. We use the globally-centered variant instead for three reasons: (1) the adaptive prior provides empirically useful regularization in label-imbalanced genomic sequences (see below); (2) the learned duration bias $\mathcal{B}_{k,c}$, with $K \times C$ free parameters, can absorb the average effect of $\nu_{b,c}$, so the prior primarily regularizes per-sample variation rather than imposing a fixed bias (parametric duration models with $O(C)$ parameters cannot represent the linear-in-k correction and likely require the reconstruction variant instead); and (3) the reconstruction variant requires propagating $\nu_{b,c}$ into the Triton kernel as an additional argument, adding complexity for marginal benefit in the fixed-length-window regime where $L_b = T$.

Label c	f_θ (active)	f_θ (inactive)	Prevalence	$\nu_{b,c}$	$-\nu_{b,c} \cdot 100$
INTRON	$+4.0$	-1.0	85%	$+3.25$	-325
EXON	$+5.0$	-0.5	14%	$+0.27$	-27
PROMOTER	$+8.0$	-0.2	1%	-0.12	$+12$

Table 3: Toy example of centering under label imbalance ($T = 10{,}000$). The baseline $\nu_{b,c}$ is the weighted average: $\nu_{b,c} = p_c \cdot f_\theta^{(\text{active})}(c) + (1 - p_c) \cdot f_\theta^{(\text{inactive})}(c)$, where p_c is the label prevalence. The dominant label (INTRON) receives a strong duration penalty (-325 at k=100), while the rare label (PROMOTER) receives a slight duration *bonus* ($+12$) because its baseline is negative. Centered emissions at active positions are $f_\theta^{(\text{active})}(c) - \nu_{b,c}$: $+0.75$ (INTRON), $+4.73$ (EXON), $+8.12$ (PROMOTER)—centering amplifies the signal-to-background contrast for rare labels.

only at the output level. The encoder can learn representations that are well-calibrated relative to the centering, and the implicit regularization provides robustness to emission scale miscalibration during early training.

- *CRF → downstream model.* When CRF marginals serve as input features for a downstream model (e.g., a state-space model), the data-dependent prior alters the posterior marginals in a sample-specific way. In this setting, a path-invariant centering (per-position, label-agnostic) is preferable to ensure that the downstream model receives marginals that reflect the canonical semi-CRF distribution. A suitable choice is $b_t = \max_c f_\theta(t, c)$, which reduces prefix-sum magnitude while preserving the model's path distribution exactly.

9.2 Additional Stability Measures

Log-domain. All computations use logsumexp: $\text{LSE}(\mathbf{x}) = \max(\mathbf{x}) + \log \sum_i \exp(x_i - \max(\mathbf{x}))$.

NEG_INF guards. When all logsumexp inputs are -10^9, the subtraction $x - \max(x) = 0$ instead of remaining at $-\infty$. Guards detect this case ($\max < -10^9 + 1$) and return $-\infty$ directly.

Float64 accumulation. Gradient tensors for shared parameters use float64 to prevent nondeterminism from atomic_add floating-point non-associativity. Error scales as $O(\sqrt{T \times K \times C})$ per operation; float64 reduces this from $\sim 10^{-3}$ (float32) to $\sim 10^{-10}$ (negligible).

Log-marginal clamping. Before exp(), log-marginals are clamped to $[-700, 700]$ to prevent float64 overflow ($\exp(710) \approx \infty$).

Masking. Invalid positions use -10^9 (not $-\infty$) to avoid NaN in gradients.

Variable-length batches. For sequences ending at $L < T$, the normalization shift is masked to zero for $t > L$, freezing $\mathcal{N}_{\text{accum}}$ at the correct value.

10 Implementation Correspondence

Math	Code Variable	Notes
$\mathcal{S}_{t,c}$	`cum_scores[:, t, c]`	Built from centered emissions
$f_\theta(t, c)$	`scores` / `projected`	Raw encoder output
$\nu_{b,c}$	`scores_float.mean(dim=1, keepdim=True)`	Mean over padded T
$\mathcal{T}_{c',c}$	`transition[c_src, c_dst]`	Source-first storage
$\mathcal{B}_{k,c}$	`duration_bias[k-1, c]`	0-indexed in code
$\boldsymbol{\alpha}$	`ring_buffer` / `alpha_ring`	Size K
$\boldsymbol{\beta}$	`beta_ring`	Size $2K$ (Section 5)
$\mathcal{P}^{\text{start}}$	`proj_start`	Optional, (B, T, C)
$\mathcal{P}^{\text{end}}$	`proj_end`	Optional, (B, T, C)
Ω	`ring_checkpoints`	Saved at intervals
$\mathcal{N}$	`log_norm_checkpoints`	Cumulative log-norm
Δ	`checkpoint_interval`	$\approx \sqrt{TK}$
C_{PAD}	`C_PAD`	`_next_power_of_2(C)`
τ	`TILE_C`	Adaptive (Section 6)
—	`grad_tr_workspace`	$(B, N, C_{\text{PAD}}, C_{\text{PAD}})$ or $(B, N, K, C_{\text{PAD}}, C_{\text{PAD}})$
—	`grad_db_workspace`	$(B, N, K, C_{\text{PAD}})$

Table 4: Mapping between mathematical notation and implementation. N denotes number of segments/checkpoints.

11 Complexity Summary

Component	Time	Memory
Forward scan	$O(TKC^2)$	$O(KC)$ ring buffer
Checkpoints	—	$O(\sqrt{T/K} \cdot KC)$
Backward scan	$O(TKC^2)$	$O(KC)$ ring buffer ($2K$ slots)
Alpha recompute	$O(TKC^2)$	$O(\Delta \cdot C)$ per segment
Gradient workspace	—	$O(B \cdot N \cdot KC^2)$
Total	$O(TKC^2)$	$O(KC + \sqrt{T/K} \cdot KC)$

Table 5: Complexity analysis. Memory is independent of T (excluding checkpoints).

References

[1] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *NeurIPS*, 2022.

[2] Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[3] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training Deep Nets with Sublinear Memory Cost. *arXiv preprint arXiv:1604.06174*, 2016.

[4] Monroe D. Donsker. An Invariance Principle for Certain Probability Limit Theorems. *Memoirs of the American Mathematical Society*, 6, 1951.

[5] E. S. Page. Continuous Inspection Schemes. *Biometrika*, 41(1–2):100–115, 1954.

[6] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ICML*, 2015.

[7] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Mądry. How Does Batch Normalization Help Optimization? *NeurIPS*, 2018.

[8] Bradley Efron. *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*. Cambridge University Press, 2010.