Block-Wise Differentiable Sinkhorn Attention: Tail-Refinement Gradients with a Single-Active-Dustbin Bridge

Dylan B. Forde
Independent Researcher
forde.dylan@gmail.com

Abstract

Sinkhorn attention balances how queries use keys, making it a useful candidate when attention should represent a shared allocation. Training it on long sequences is difficult because automatic differentiation can retain large intermediate transport matrices. We derive an exact backward pass for a simple training rule: run a base solve without gradients, then differentiate a short refinement tail. For two refinement steps, the backward pass uses four transport factors. We show that all four can be reconstructed from one reference tile and row and column scaling vectors. This gives two exact schedules, direct recomputation and one-reference evaluation, with linear auxiliary storage on fixed-width bands. At length $16{,}384$ on one TPU, one-reference uses 34.7 MiB of compiler-accounted memory, compared with 578.5 MiB for four-plan retention and 3204.1 MiB for automatic differentiation. Both low-state schedules complete exact surrogate backward passes at $262{,}144$ positions per device. Direct recomputation is currently faster. Protein reconstruction experiments demonstrate trainability, but favor trained FlashAttention in a matched short-run comparison and do not establish alignment superiority. The contribution is an exact, memory-efficient way to differentiate the stated Sinkhorn surrogate, together with a precise account of its limits.

1 Introduction

Attention builds an output by weighting and combining values. In standard softmax attention, each query chooses its weights independently: several queries can place most of their weight on the same key [Vaswani et al., 2017]. This is often desirable. In a matching problem, however, one may also want to control how much total weight each key receives. Sinkhorn attention introduces this second constraint. It alternately normalizes rows and columns, coupling the choices made by different queries through a shared allocation [Cuturi, 2013, Peyré and Cuturi, 2019].

This distinction matters when attention is intended to express a correspondence, rather than only retrieve useful content. For example, matching positions between two protein sequences calls for reasoning about which positions correspond and which compete for the same partner. Balanced transport offers one way to encode that competition. It does not, by itself, supply the right features, loss, or treatment of gaps. Our protein experiments test this motivation and retain its limitations: learning to reconstruct features did not produce strong positional alignment.

The computational difficulty appears during training. A Sinkhorn solve repeatedly updates the attention weights. A straightforward backward pass can store intermediate matrices across these iterations, making long sequences expensive even when the forward computation is tiled. Restricting attention to a band reduces the number of allowed query–key pairs, but does not by itself determine what must be saved for differentiation.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

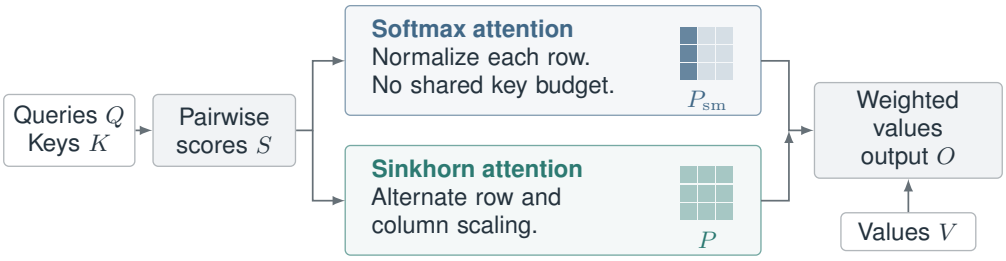

Figure 1: Softmax normalizes rows independently; Sinkhorn also allocates shared column capacity. These schematic weights illustrate square, compatible unit targets. Finite Sinkhorn iterates need not satisfy both targets exactly.

We study a concrete alternative. First, run T Sinkhorn iterations to obtain a base state and stop gradients through that computation. Then run R additional iterations and differentiate those refinements exactly. This training rule lets a longer forward solve prepare the weights while keeping the differentiated computation short. Its gradient belongs to this *stopped-base surrogate*; it need not equal the gradient through all $T + R$ iterations.

Our main result concerns the choice $R = 2$ used in the reported implementation. Differentiation produces four transport factors involving row and column scalings from different refinement steps. These factors look like four matrices to compute or retain. In fact, they all share the same scores, and differ only by row and column multipliers. One reference tile therefore contains the matrix information needed for all four. The remaining information fits in vectors.

This structure gives two exact implementations of the same derived backward pass. *Direct recomputation* forms the four factors inside each tile. *One-reference evaluation* forms one factor and reconstructs the other contributions using vector multipliers. Both avoid retaining full transport plans. The latter reduces the number of nonlinear plan constructions, although our current compiler executes direct recomputation faster. The distinction between an algebraic saving and a measured speed improvement is central to interpreting the results.

We also give conditional bounds on the gradient omitted by stopping the base solve, and extend the adjoint to an augmented problem with one active dustbin token per side. Our experiments assess numerical correctness, memory, runtime, and protein reconstruction separately. Both exact schedules run at $262{,}144$ positions per device. The reconstruction and alignment studies then test the limits of what this computational result implies for task quality.

Code availability. `github.com/dylanbforde/Papers/tree/main/block-wise-sinkhorn`.

Relation to prior work. Sinkhorn iteration is a standard way to compute entropically regularized transport [Cuturi, 2013, Peyré and Cuturi, 2019]. Differentiating transport and iterative layers has also been studied through analytic and implicit derivatives [Luise et al., 2018, Bai et al., 2019]. Our object is the finite stopped-base attention map and its arbitrary-output-gradient backward pass. Sparse attention reduces the interaction set [Beltagy et al., 2020, Zaheer et al., 2020]; memory-aware attention kernels reduce transfers and retained intermediates [Dao et al., 2022]. We use both a fixed band and tiling, then exploit the additional structure of the Sinkhorn adjoint. FlashSinkhorn [Ye et al., 2026] provides a dense transport implementation and OT-cost derivatives. Its validated interface does not expose the same hard-band attention-output backward pass, so our comparison with it is a dense-forward backend diagnostic (Section 5).

2 From Softmax Attention to Sinkhorn Attention

We describe one attention head and omit batch indices. Let $Q, K, V \in \mathbb{R}^{L\times d}$ be queries, keys, and values. As in scaled dot-product attention, form compatibility scores $A = QK^\top/\sqrt{d}$. We write $S = A/\varepsilon$ for scores divided by a positive temperature ε. The temperature controls how sharply weights favor high scores. Let Ω be the fixed set of allowed query–key pairs; disallowed pairs

Table 1: Notation used in the main derivation. Symbols are introduced where needed; Appendix D gives the full glossary.

Symbol	Meaning
Q, K, V; O	Queries, keys, values; attention output.
L, d, W, Ω	Sequence length, head dimension, band width, allowed pairs.
A; $S = A/\varepsilon$	Dot-product scores; temperature-scaled scores.
u, v; P	Log row and column scalings; resulting attention weights.
T; R	Base iterations with stopped gradients; differentiated tail iterations.
G; $\bar{S}$	Incoming output gradient; gradient with respect to S.

always have zero weight. The band parameter W limits each query to $O(W)$ nearby keys; the implementation uses a half-band convention.

Both attention operators have the form $O = PV$, where P is the matrix of weights. Softmax normalizes each row of $\exp(S)$ once. Sinkhorn repeatedly rescales both rows and columns. In the square unit-target case, its aim is

$$P\mathbf{1} = \mathbf{1}, \qquad P^\top \mathbf{1} = \mathbf{1}. \tag{1}$$

Each query then distributes one unit of weight, and each key receives one unit in total. These constraints describe a fractional allocation, not a discrete one-to-one match. They require support compatible with the targets; a fixed number of iterations need not satisfy both exactly.

2.1 Balancing with two vectors

The weights need not be stored as a matrix. Two vectors u and v encode the row and column rescalings in log space:

$$P_{ij} = \exp(S_{ij} + u_i + v_j) \quad \text{for } (i,j) \in \Omega, \qquad P_{ij} = 0 \quad \text{otherwise.} \tag{2}$$

Adding u_i multiplies row i by e^{u_i}; adding v_j multiplies column j by e^{v_j}. Log space permits stable evaluation by log-sum-exp. A full Sinkhorn iteration first adjusts every row and then every column:

$$u_i^{(t+1)} = -\log \sum_{j:(i,j)\in\Omega} \exp(S_{ij} + v_j^{(t)}), \tag{3}$$

$$v_j^{(t+1)} = -\log \sum_{i:(i,j)\in\Omega} \exp(S_{ij} + u_i^{(t+1)}). \tag{4}$$

The first update makes the current row sums one. The second makes the column sums one, generally changing the row sums again. Repeating the pair seeks agreement between the two sets of constraints. The implementation streams score tiles to compute these reductions and retains the scaling vectors.

The output is the weighted value sum

$$O_i = \sum_{j:(i,j)\in\Omega} P_{ij} V_j. \tag{5}$$

We apply no extra row normalization to this training output. At finite depth, it can therefore differ from a convex average. Row-normalized weights are used only for the positional diagnostics described later.

Rectangular sequences. The same finite updates can be evaluated for n_q query positions and n_k key positions. Their unit row and column targets have total masses n_q and n_k. If these differ, both targets cannot be satisfied simultaneously. Our rectangular Pfam implementation uses these finite *unit-target updates*; it is not a converged balanced rectangular OT solve. General rectangular balanced transport requires compatible marginals of equal total mass. The finite map and its derivative remain well defined for finite active scores and nonempty active row and column neighborhoods.

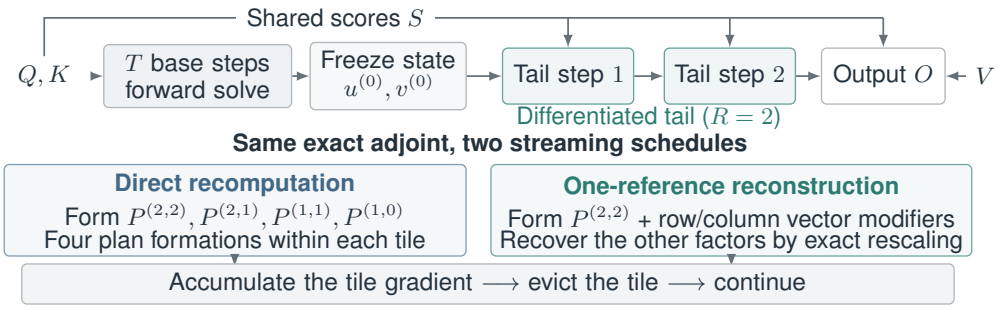

Figure 2: Both schedules give the exact adjoint of the fixed-base $R = 2$ surrogate. One-reference uses a single plan tile and vector modifiers; no latency advantage follows.

3 An Exact Backward Pass with One Reference Tile

3.1 Differentiate only the refinement tail

The forward pass first performs T iterations of equations (3)–(4). We treat the resulting pair $(u^{(0)}, v^{(0)})$ as constant during differentiation, and number the next R iterations from 1 to R. Their final weights define the output through equation (5). Figure 2 shows this boundary.

An incoming gradient $G = \partial\mathcal{L}/\partial O$ tells the layer how a scalar loss $\mathcal{L}$ changes with its output. Our backward pass returns the corresponding gradients of Q, K, V for any G. This is the vector–Jacobian product (VJP) required to place the layer inside a larger trainable model; it is not restricted to a particular transport cost or reconstruction loss. Below, a bar denotes a gradient accumulated in this reverse pass, so $\bar{S} = \partial\mathcal{L}/\partial S$.

There are two separate correctness questions. The custom backward should agree with automatic differentiation of the *same stopped-base computation*. We derive this equality exactly and test its numerical implementation. Whether stopping the base gradient is a good approximation to differentiating the full solve is a different question, addressed in Section 4.

3.2 Why four factors share one reference

During the reverse pass, the row update and column update each contribute a transport factor. For $R = 2$, there are four:

$$P^{(2,2)}, \quad P^{(2,1)}, \quad P^{(1,1)}, \quad P^{(1,0)}.$$

The first superscript selects the row-scaling vector and the second selects the column-scaling vector: $P_{ij}^{(a,b)} = \exp(S_{ij} + u_i^{(a)} + v_j^{(b)})$ on the support. Thus $P^{(2,1)}$ combines the second row update with the first column update. These mixed-time factors arise from differentiating the successive normalizations; they are not four separate forward attention layers.

Proposition 1 (One-reference backward)**.** *Fix the support with nonempty active row and column neighborhoods, finite active scores, and stopped base state, and take $R = 2$. In raw log-scaling coordinates, let $P_{\mathrm{ref}} = P^{(2,2)}$. For each of the four pairs (a, b) above and every $(i, j) \in \Omega$,*

$$P_{ij}^{(a,b)} = P_{\mathrm{ref},ij} \underbrace{\exp(u_i^{(a)} - u_i^{(2)})}_{\textit{row multiplier}} \underbrace{\exp(v_j^{(b)} - v_j^{(2)})}_{\textit{column multiplier}}. \tag{6}$$

The exact gradient for any incoming G can consequently be evaluated using one reference plan tile at a time, row and column vector slices, and the required slices of Q, K, V, G. No other full transport factor needs to be retained.

Reasoning. All four factors contain the same $\exp(S_{ij})$. Dividing out the reference leaves only differences in the row and column scalings, giving equation (6). Differentiating each log-sum-exp normalization yields weighted row or column reductions. Applying the identity to those reductions and to the score gradient removes the need to store the other three factors. Appendix A gives the complete reverse recurrence, the expanded score-gradient formula, and the proof. The scaling identity also holds for longer tails; $R = 2$ gives the compact four-factor backward evaluated here.

This explains the saving: a tile contains pairwise information, while a row or column multiplier contains only one number per position. Once the score-gradient tile is formed, it contributes to ordinary matrix-product gradients:

$$\bar{Q} = \frac{\bar{S}K}{\sqrt{d}\,\varepsilon}, \qquad \bar{K} = \frac{\bar{S}^\top Q}{\sqrt{d}\,\varepsilon}, \qquad \bar{V} = P_{\mathrm{ref}}^\top G. \tag{7}$$

These products are accumulated tile by tile; the displayed matrices need not be materialized globally.

Numerical representation. The implementation centers the log scalings to keep their magnitudes controlled. Adding a constant to v and subtracting it from u leaves a same-time plan unchanged. When vectors from different times are combined, their centering constants must also be accounted for. Proposition 1 uses raw coordinates; Appendix A records the scalar corrections needed for stored centered coordinates. This bookkeeping is necessary for exactness.

3.3 What is saved, and what is recomputed

The two schedules use the same adjoint. Direct recomputation evaluates each of the four factors from the score tile and its scaling vectors. One-reference evaluation constructs P_{ref} once and applies the multipliers in equation (6). The latter replaces three nonlinear plan constructions with pointwise operations. Either schedule can release the tile after accumulating its contributions, and repeated tile sweeps perform the required global vector reductions. The full scheduling details appear in Appendix E.

For a tile of size $B \times B$, one-reference evaluation keeps one $B \times B$ plan tile, $O(B)$ scaling and gradient-vector slices, and $O(Bd)$ activation slices. Across the sequence, inputs and returned gradients occupy $O(Ld)$ storage, while the auxiliary vector state is $O(L)$ for fixed iteration depths. For fixed d and B, a T-step base and R-step tail require $O((T+R)LW)$ arithmetic on a band. Thus the linear dependence on L assumes fixed W, T, R, d, B; dense support $W = L$ retains quadratic interaction work.

These are algorithmic storage and work bounds. Compiler-accounted memory includes other buffers, and hardware latency depends on fusion, recomputation, and memory traffic. We measure both quantities rather than infer one from the other.

4 Approximation and Scope

What stopping the base gradient omits. Changing the scores changes both the refinement steps and their starting state. The surrogate differentiates the first dependence and omits the second. If refinement is locally contractive, later iterations become less sensitive to their starting state, so a longer differentiated tail can reduce this omission. Appendix B makes this conditional statement precise: under its fixed-support smoothness and full-state contraction assumptions, the output-Jacobian discrepancy is bounded by a constant times ρ^R, where $0 \le \rho < 1$. A corresponding bound holds for scalar-loss gradients under the stated loss-derivative bound.

The appendix also gives an a posteriori certificate based on the gradient arriving at the stopped state, and a rule for choosing the cheapest tested depth whose certified error is within a tolerance. These results do not assert that a two-step tail is accurate for every input. The projective contraction analysis applies to strictly positive active blocks; it does not automatically establish the full-state contraction assumption on an arbitrary masked problem. The reported local checks illustrate the certificate, while the long-context measurements test exact evaluation of the chosen surrogate.

A limited way to represent unmatched positions. A dustbin is an additional token that can receive or supply weight when a position has no suitable partner. Our implementation adds one active dustbin on each side, with masked filler entries for block alignment. Applying the same unit-target updates to this augmented support preserves the adjoint and one-reference identity (Appendix B). Each dustbin has one unit of target capacity; this is not a general model of arbitrary insertion or deletion mass. The current code also widens the band to connect the dustbins, giving dense interaction work when $W < L$. The fixed-band long-context bound applies to the banded path, not this widened implementation.

Implementation	Complete step (ms)	Compiler total (MiB)
Literal four-plan retention	257.4	578.5
Direct recomputation	260.1	34.7
One-reference	334.9	34.7
Exact autodiff	393.2	3204.1

Table 2: Complete-step medians at $L = 16384$ on one V5E1 TPU. Compiler total counts executable arguments, outputs, and temporaries, minus aliases. It is a static estimate, not a measured runtime peak.

5 Experiments

We test three questions: whether the derived backward agrees with autodiff of the same surrogate, how its implementation changes state and runtime, and what the resulting attention layer learns. Appendix C contains the full results; Appendix C.9 gives the numerical, hardware, training, and evaluation protocols.

Correctness of the implemented backward. We compare literal four-plan retention, direct recomputation, one-reference evaluation, and exact autodiff through the stopped-base surrogate. Across the single-device TPU comparisons at lengths 2048, 8192, and 16384, the maximum absolute output and $Q/K/V$-gradient discrepancies are 4.47×10^{-8} and 2.85×10^{-5}, respectively. Small masked problems provide additional gradient and surrogate-bias checks. These numerical checks test implementations of the same differentiated object; the difference from differentiating through the base solve is assessed separately by the bias certificates.

State savings and runtime are separate outcomes. Table 2 compares all four implementations on the same V5E1 TPU. Every timing includes the forward pass, saved-state creation and retention, and the backward pass for an arbitrary output gradient. Each row uses $T = 15$, $R = 2$, $W = 1024$, float32 inputs, five warmups, and twenty synchronized measurements.

The compiler audit verifies that literal retention keeps and consumes four distinct plans. Its explicit saved-plan payload is 544 MiB, compared with 0.125 MiB of vector state for each low-state schedule; these payloads are already included in the compiler totals. Direct recomputation and one-reference are both exact schedules derived in this paper. The one-reference identity forms fewer plans, but the present compiler makes direct recomputation faster with essentially equal memory accounting. Literal retention is slightly faster still at this length. The result supports the low-state adjoint family and its factorization, without establishing a universal latency winner. Independently measured dynamic peak HBM was unavailable.

The complete backward also runs at extreme context. Both low-state schedules complete the exact surrogate VJP at 262144 positions per device on all eight v6e devices. Each device processes a distinct full sequence, without sequence sharding. Complete-step medians are 6.020 s for one-reference and 5.415 s for direct recomputation, with compiler totals 835.715 and 835.528 MiB. These BF16-input runs use one warmup and three measurements. The separate-allocation comparison with literal retention is detailed in Appendix C. The compiler's eight-device accounting scope is not established as either per-device or aggregate. The explicit value-and-VJP interface removes an earlier lowering-specific compilation boundary, with each method matching its own earlier output and gradient hashes at 16384; 262144 is a tested context, not a maximum feasible length.

Baseline interfaces determine the comparison. Native dense FlashAttention and same-support Local SplashAttention also complete at 262144. They provide capacity references: dense attention changes support, while local attention computes softmax instead of transport. The validated official FlashSinkhorn interface does not expose our hard-band $Q/K/V$ backward for arbitrary output gradients or a matched Pfam training surface. On its supported dense-forward A100 surface, official FlashSinkhorn is faster than our forced-dense, non-native execution. The full diagnostic and its execution-policy differences are reported in Appendix C.

Seed	Sinkhorn MSE	FlashAttention MSE
0	0.054595	0.052842
1	0.052788	0.050680
2	0.052374	0.050782
Mean	0.053252	0.051435

Table 3: Matched 100-update Pfam reconstruction screen; lower MSE is better. FlashAttention has lower error in every seed pair. These are short-horizon validation results, not convergence evidence.

Matched training favors FlashAttention on reconstruction. We use paired protein sequences from Pfam [Mistry et al., 2021], represented by ESM-2 embeddings [Lin et al., 2023], with learned query/key projections and a fixed value projection. At each annotated query position, reconstruction compares the attention output with the value vector at its known partner position. Positional metrics instead ask whether the weights identify that partner. The Pfam screen holds examples, trainable projections, initialization policy, optimizer, and update budget fixed across methods. Each seed receives 100 updates and evaluation on the same 5000 validation examples. Training uses native BF16 TPU kernels; FlashAttention quality is evaluated using a dense float32 reference on the trained projections.

The mean paired Sinkhorn-minus-FlashAttention difference is $+0.0018174$. Three seeds provide descriptive evidence for this fixed protocol. All rows use the V5E1 hardware class, without proven physical-device identity; the recovered Sinkhorn seed-2 row supports reconstruction results only. The screen supplies neither a timing comparison nor task superiority for Sinkhorn.

Trainability does not imply positional alignment. A separate longer Sinkhorn run reaches step 1437 and reduces reconstruction error by 63.1% on its 100 held-out test examples, from 5.5730 to 2.0539. Its positional alignment remains much weaker than a deterministic diagonal reference. The error of its average predicted target position (the transport barycenter) is 64.59 versus 1.64 columns; within-five accuracy is 4.03% versus 93.72%. This evaluation differs from the short-screen validation set above.

Direct alignment supervision does not resolve this limitation in the prespecified short calibration. At seed 0, cross-entropy (CE) weights 0.1, 0.3, and 1.0 produce tiny improvements in CE, barycenter error, and reconstruction, but slightly reduce within-five accuracy. No weight passes the fixed step-100 selector, so no weight or checkpoint is selected and seed-$1/2$ replication is not triggered. The empirical contribution is therefore exact long-context backward execution and demonstrated trainability; improved alignment or downstream performance remains open.

6 Discussion

The result makes a trainable transport operator available at lengths where retaining intermediate plans is costly. A model that needs query-side and key-side allocation can use the exact stopped-base backward with vector state and streamed tiles. The one-reference identity also specifies a concrete target for specialized backends: one plan construction, vector modifiers, and reductions. The current advantage over direct recomputation is structural; translating it into lower latency remains a compiler and kernel question.

Balancing constrains the use of keys, but useful correspondences still depend on the representation, supervision, and treatment of unmatched positions. The Pfam experiments expose a reconstruction–alignment gap despite successful training. Longer matched runs, stronger alignment objectives, and evaluation in another domain would test when this allocation structure helps. Our present evidence establishes the exact adjoint, its low-state execution, and a long-context operating point, while leaving those task-level benefits open.

Broader impact. Reducing retained attention state may make experiments with structured attention less memory intensive. This potential benefit is conditional on how the operator is used: lower memory requirements can also enable larger workloads, and we have not measured net energy savings. The method is a general component and does not prevent harmful downstream applications. The protein experiments do not validate biological or clinical use.

References

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. doi: 10.48550/arXiv.2004.05150.

Garrett Birkhoff. Extensions of Jentzsch's theorem. *Transactions of the American Mathematical Society*, 85(1):219–227, 1957. doi: 10.1090/S0002-9947-1957-0087058-6.

P. J. Bushell. Hilbert's metric and positive contraction mappings in a Banach space. *Archive for Rational Mechanics and Analysis*, 52(4):330–338, 1973. doi: 10.1007/BF00247467.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html`.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023. doi: 10.1126/science.ade2574.

Giulia Luise, Alessandro Rudi, Massimiliano Pontil, and Carlo Ciliberto. Differential properties of Sinkhorn approximation for learning with Wasserstein distance. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Jaina Mistry, Sara Chuguransky, Lowri Williams, Matloob Qureshi, Gustavo A Salazar, et al. Pfam: The protein families database in 2021. *Nucleic Acids Research*, 49(D1):D412–D419, 2021. doi: 10.1093/nar/gkaa913.

Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019. doi: 10.1561/2200000073.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL `https://arxiv.org/abs/1706.03762`.

Felix X.-F. Ye, Xingjie Li, An Yu, Ming-Ching Chang, Linsong Chu, and Davis Wertheimer. FlashSinkhorn: IO-aware entropic optimal transport. *arXiv preprint arXiv:2602.03067*, 2026. doi: 10.48550/arXiv.2602.03067.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297, 2020.

A Exact Fixed-Depth Adjoint and Proofs

This appendix gives the full definitions, adjoint identities, coordinate ledger, and proofs summarized in the main text. All reverse derivatives refer to the stated finite-depth surrogate.

A.1 Forward Pass: Block-Wise Balanced Sinkhorn

Let Q, K, V be query, key, and value tensors with head dimension d, and let Ω denote a fixed active support with band width W. We absorb the entropic temperature ε into the score matrix

$$S_{ij} = \frac{Q_i K_j^\top}{\sqrt{d}\,\varepsilon}.$$

The stopped base solve and the refinement tail both use the same masked balanced half-steps

$$u_i^{(t+1)} = -\log\Big[\sum_{j:(i,j)\in\Omega} \exp(S_{ij} + v_j^{(t)})\Big], \tag{8}$$

$$v_j^{(t+1)} = -\log\Big[\sum_{i:(i,j)\in\Omega} \exp(S_{ij} + u_i^{(t+1)})\Big]. \tag{9}$$

These are unit-target updates: each active row and active column target mass is one. For n_q active rows and n_k active columns, their total target masses are n_q and n_k. Equal totals require $n_q = n_k$; general rectangular balanced OT instead needs specified marginals a, b satisfying $\sum_i a_i = \sum_j b_j$, together with support compatible with those marginals. On rectangular masked Pfam examples, the code still applies the displayed unit-target updates. We call this the implemented rectangular unit-target surrogate. For finite active scores and nonempty active row and column neighborhoods, its finite-depth map and adjoint are well defined, but when $n_q \neq n_k$ its unit targets cannot both be satisfied by a balanced transport plan. The implementation computes these updates in tiles, in the same spirit as IO-aware attention kernels. For fixed band width W, a T-step base solve followed by an R-step tail has cost $O((T+R)LW)$ time, $O(Ld)$ input storage, and $O(L)$ additional asymptotic working storage.

Remark 1 (Fixed-Budget ε-Scaling)**.** *The implementation optionally uses a short continuation schedule $\varepsilon_1 \geq \cdots \geq \varepsilon_S = \varepsilon$ during the stopped base solve. If the stage iteration counts satisfy $\sum_{s=1}^{S} T_s = T$, the asymptotic cost remains $O(TLW)$ time and $O(L)$ additional working storage for fixed head dimension.*

A.2 Balanced Tail Refinement and the $R = 2$ Staircase

Let $(u^{(0)}, v^{(0)})$ be the stopped potentials produced by the base T-step solve. The differentiated object is not the full T-step map; it is the fixed-depth surrogate

$$\widetilde{O}^{[R]}(Q, K, V; u^{(0)}, v^{(0)}) = \mathcal{A}(Q, K, V, u^{(R)}, v^{(R)}),$$

where the refinement iterates are recomputed from the stopped base pair. The implementation stores centered dual representatives for numerical stability, while the main algebra is written in ungauged coordinates. The centered gauge ledger below makes the coordinate convention explicit: same-time plans are gauge-invariant, while mixed-time centered plans require explicit scalar gauge factors when translated back to ungauged coordinates. The terminal transport application is

$$\mathcal{A}(Q, K, V, u, v)_i = \sum_{j:(i,j)\in\Omega} P_{ij}(u, v) V_j, \qquad P_{ij}(u, v) = \exp(S_{ij} + u_i + v_j).$$

There is no extra row renormalization in the training output. Reconstruction uses this raw transport output, sparse CE logs the target-entry transport mass, and row-normalized barycenters are used only for alignment diagnostics and visualization.

Define staircase plans

$$P_{ij}^{(a,b)} = \begin{cases} \exp(S_{ij} + u_i^{(a)} + v_j^{(b)}), & (i,j) \in \Omega, \\ 0, & (i,j) \notin \Omega. \end{cases}$$

Let $\mathcal{L}$ be a scalar loss depending on $\widetilde{O}^{[R]}$, let $G = \partial\mathcal{L}/\partial\widetilde{O}^{[R]}$, and define

$$Z_{ij} = \langle G_i, V_j \rangle, \qquad g_u = (P^{(R,R)} \odot Z)\mathbf{1}, \qquad g_v = (P^{(R,R)} \odot Z)^\top \mathbf{1}.$$

Proposition 2 (Exact Fixed-Depth Tail Adjoint)**.** *Fix $R \geq 1$, finite active scores, and a fixed support with nonempty active row and column neighborhoods. The cotangents of the refinement duals satisfy*

$$\bar{v}^{(R)} = g_v, \tag{10}$$

$$\bar{u}^{(R)} = g_u - P^{(R,R)}\bar{v}^{(R)}, \tag{11}$$

$$\bar{v}^{(t-1)} = -\big(P^{(t,t-1)}\big)^\top \bar{u}^{(t)}, \qquad t = R, R-1, \ldots, 1, \tag{12}$$

$$\bar{u}^{(t-1)} = -P^{(t-1,t-1)}\bar{v}^{(t-1)}, \qquad t = R, R-1, \ldots, 2, \tag{13}$$

where the final line stops at $t = 2$ because the base state is stop-gradient. The score-space cotangent is

$$\bar{S} = P^{(R,R)} \odot Z - \sum_{t=1}^{R} \Big[P^{(t,t)} \odot \big(\mathbf{1}\bar{v}^{(t)\top}\big) + P^{(t,t-1)} \odot \big(\bar{u}^{(t)}\mathbf{1}^\top\big) \Big]. \tag{14}$$

For the reported TPU kernel we use $R = 2$, which yields the staircase

$$\big(P^{(2,2)},\, P^{(2,1)},\, P^{(1,1)},\, P^{(1,0)}\big).$$

Proposition 3 (One-Reference-Tile $R = 2$ Backward Schedule)**.** *For any staircase pair $(a, b) \in \{(2,2), (2,1), (1,1), (1,0)\}$,*

$$P^{(a,b)} = P^{(2,2)} \odot \exp(u^{(a)} - u^{(2)}) \odot \exp(v^{(b)} - v^{(2)}).$$

Define

$$\alpha_i = \exp(u_i^{(1)} - u_i^{(2)}), \qquad \beta_j = \exp(v_j^{(1)} - v_j^{(2)}), \qquad \delta_j = \exp(v_j^{(0)} - v_j^{(2)}).$$

Then on the active support the exact $R = 2$ score-space VJP is

$$\bar{S}_{ij} = P_{ij}^{(2,2)} \Big[Z_{ij} - \bar{v}_j^{(2)} - \beta_j \bar{u}_i^{(2)} - \alpha_i\beta_j\bar{v}_j^{(1)} - \alpha_i\delta_j\bar{u}_i^{(1)} \Big].$$

Hence, for a $B \times B$ score tile, the exact backward can be scheduled with one resident reference plan tile $P_{I,J}^{(2,2)}$, the corresponding $O(B)$ vector slices, and the $O(Bd)$ activation/cotangent slices needed to form $S_{I,J}$ and $Z_{I,J}$. The remaining staircase factors are never formed as resident plan tiles. For fixed block size, head dimension, band width, T, and $R = 2$, the schedule uses $O(Ld)$ input storage, $O(L)$ global vector state, and $O(LW)$ tile arithmetic for the tail backward.

Remark 2 (Centered gauge ledger)**.** *The displayed formula is the ungauged statement. If the stored centered representatives satisfy*

$$\widetilde{u}^{(t)} = u^{(t)} - c_t\mathbf{1}, \qquad \widetilde{v}^{(t)} = v^{(t)} + c_t\mathbf{1},$$

then same-time plans are unchanged but mixed-time plans are not. With

$$\widetilde{\alpha}_i = \exp(\widetilde{u}_i^{(1)} - \widetilde{u}_i^{(2)}), \quad \widetilde{\beta}_j = \exp(\widetilde{v}_j^{(1)} - \widetilde{v}_j^{(2)}), \quad \widetilde{\delta}_j = \exp(\widetilde{v}_j^{(0)} - \widetilde{v}_j^{(2)}),$$

the ungauged mixed factors recovered from the centered ledger are

$$P_{ij}^{(2,1)} = e^{c_2 - c_1} P_{ij}^{(2,2)} \widetilde{\beta}_j, \quad P_{ij}^{(1,1)} = P_{ij}^{(2,2)} \widetilde{\alpha}_i \widetilde{\beta}_j, \quad P_{ij}^{(1,0)} = e^{c_1 - c_0} P_{ij}^{(2,2)} \widetilde{\alpha}_i \widetilde{\delta}_j.$$

The proof section below records this explicitly because the mixed-time factors are not gauge-invariant.

For $R = 2$, equation (14) expands to

$$\begin{aligned} \bar{S} = {} & P^{(2,2)} \odot Z - P^{(2,2)} \odot \big(\mathbf{1}\bar{v}^{(2)\top}\big) - P^{(2,1)} \odot \big(\bar{u}^{(2)}\mathbf{1}^\top\big) \\ & - P^{(1,1)} \odot \big(\mathbf{1}\bar{v}^{(1)\top}\big) - P^{(1,0)} \odot \big(\bar{u}^{(1)}\mathbf{1}^\top\big). \end{aligned} \tag{15}$$

The reconstruction identity above then replaces separate formation of the last three plan tiles with row and column rescalings of $P^{(2,2)}$.

Figure 3 summarizes the stop-gradient boundary and tile-level memory schedule. Each score block forms one reference plan tile, applies row and column vector modifiers for the other three factors, accumulates the score cotangent, and then evicts the tile.

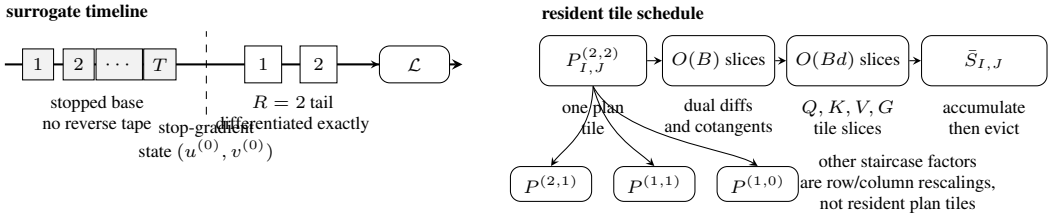

Figure 3: Stopped-base surrogate and $R = 2$ streaming backward schedule. Left: the T-step base solve provides a stop-gradient state and only the two tail refinements are differentiated. Right: each active score tile materializes one reference plan tile $P^{(2,2)}_{I,J}$, combines it with $O(B)$ vector slices and $O(Bd)$ activation/cotangent slices, accumulates the score cotangent, and evicts the tile; the remaining staircase factors are recovered by row/column rescaling.

Why $R = 2$. The multiplicative orbit identity itself is not unique to $R = 2$. Relative to any chosen base staircase plan, every other staircase plan factors by the corresponding dual differences in the same way. What is special about $R = 2$ is that the exact adjoint closes into the compact four-factor formula above, so the identity yields a concrete one-reference-tile schedule rather than a larger family of rescaled plans.

Two exact schedules from the same adjoint. Direct recomputation evaluates the four derived factors within each tile; one-reference instead forms one plan and obtains the remaining factors by row/column rescaling. Both schedules follow from our analysis and compute the same arbitrary-cotangent vector–Jacobian product (VJP). The one-reference identity reduces nonlinear plan formation from four plans to one per active tile, but this operation count does not determine compiled latency. Our current JAX/TPU lowering favors direct recomputation in complete-step timing, with essentially equal compiler-accounted memory for the two schedules. The contribution is therefore the exact adjoint family and its stronger one-reference structure; a faster one-reference backend remains an implementation question.

A.3 Proofs and centered implementation

Notation. For the appendix proofs we work with ungauged representatives satisfying the raw masked logsumexp half-step equations. Let Ω denote the active support and define

$$P^{(a,b)}_{ij} = \begin{cases} \exp(S_{ij} + u^{(a)}_i + v^{(b)}_j), & (i,j) \in \Omega, \\ 0, & (i,j) \notin \Omega. \end{cases}$$

$$u^{(t+1)}_i = -\log\Big[\sum_{j:(i,j)\in\Omega} \exp(S_{ij} + v^{(t)}_j)\Big], \tag{16}$$

$$v^{(t+1)}_j = -\log\Big[\sum_{i:(i,j)\in\Omega} \exp(S_{ij} + u^{(t+1)}_i)\Big]. \tag{17}$$

Centering map. Let $\mathbf{1}_q$ and $\mathbf{1}_k$ denote the all-ones vectors on valid queries and keys, and let

$$m = \frac{\mathbf{1}_q}{\mathbf{1}_q^\top \mathbf{1}_q}.$$

The implementation stores centered representatives via

$$\mathcal{C}(u, v) = \big(u - (m^\top u)\mathbf{1}_q,\ v + (m^\top u)\mathbf{1}_k\big).$$

Lemma 1 (Centering Pullback)**.** *If $(\bar{u}, \bar{v})$ are cotangents on the outputs of $\mathcal{C}$, then*

$$\mathcal{C}^*(\bar{u}, \bar{v}) = \Big(\bar{u} + m\big(\mathbf{1}_k^\top \bar{v} - \mathbf{1}_q^\top \bar{u}\big),\ \bar{v}\Big).$$

Proof. Write $s = m^\top u$, so $\delta s = m^\top \delta u$. Then

$$\delta\mathcal{C}(u, v) = \big(\delta u - (\delta s)\mathbf{1}_q,\ \delta v + (\delta s)\mathbf{1}_k\big).$$

Pairing this variation with $(\bar{u}, \bar{v})$ and collecting coefficients of δu and δv yields the stated pullback. □

Lemma 2 (Gauge Equivariance)**.** *Let $\Gamma_c(u, v) = (u - c\mathbf{1}_q,\ v + c\mathbf{1}_k)$. Then the raw two-half-step map defined by equations* (16)–(17) *is gauge equivariant.*

Proof. Adding c to every valid key potential shifts every logsumexp argument in the u-update by the same constant, so the update subtracts c from every valid query potential. The v-update is identical with rows and columns swapped. □

Proposition 4 (Centered Gauge Ledger)**.** *Let $(\widehat{u}^{(t)}, \widehat{v}^{(t)})$ be the ungauged iterates and $(u^{(t)}, v^{(t)})$ the iterates produced by the centered implementation, both started from the same centered base pair. Then for every $t \geq 1$ there exists a scalar c_t such that*

$$(u^{(t)}, v^{(t)}) = \Gamma_{c_t}\big(\widehat{u}^{(t)}, \widehat{v}^{(t)}\big).$$

Consequently same-time plans are gauge-invariant,

$$\exp(S_{ij} + u_i^{(t)} + v_j^{(t)}) = \exp(S_{ij} + \widehat{u}_i^{(t)} + \widehat{v}_j^{(t)}),$$

but mixed-time staircase plans carry scalar gauge factors:

$$P^{(a,b)}_{\text{cent}\ ij} = e^{c_b - c_a}\, P^{(a,b)}_{\text{ung}\ ij}.$$

Equivalently,

$$P^{(a,b)}_{\text{ung}\ ij} = e^{c_a - c_b}\, P^{(a,b)}_{\text{cent}\ ij}.$$

The mixed factors are therefore not gauge-invariant unless $a = b$ or $c_a = c_b$.

Proof. The existence of c_t holds at $t = 1$ by definition of the centering step. Inductively, if the two previous representatives differ by a gauge transform, then gauge equivariance of the raw refinement step shows that the next raw outputs also differ by a gauge transform; applying $\mathcal{C}$ merely chooses another representative of the same orbit. The same-time identity follows because the two gauge shifts cancel inside $u_i^{(t)} + v_j^{(t)}$. For mixed times,

$$u_i^{(a)} + v_j^{(b)} = \widehat{u}_i^{(a)} + \widehat{v}_j^{(b)} - c_a + c_b,$$

which gives the scalar factor. □

Lemma 3 (When Centering Pullbacks Vanish)**.** *For cotangents $(\bar{u}, \bar{v})$ on the outputs of $\mathcal{C}$, if*

$$\mathbf{1}_q^\top \bar{u} = \mathbf{1}_k^\top \bar{v},$$

then $\mathcal{C}^*(\bar{u}, \bar{v}) = (\bar{u}, \bar{v})$. In the unit-target tail reverse pass, this equal-total condition holds at every full-step centering boundary.

Proof. The first claim is immediate from the displayed formula for $\mathcal{C}^*$. For the tail reverse pass, the terminal transport application gives

$$\mathbf{1}_q^\top g_u = \sum_{ij} P_{ij}^{(R,R)} Z_{ij} = \mathbf{1}_k^\top g_v,$$

so the final centering pullback contributes no extra term. After pulling $\bar{v}^{(t)}$ through a same-time v-half-step, the same-time plan has unit column sums, hence

$$\mathbf{1}_q^\top \Big[-P^{(t,t)} \bar{v}^{(t)}\Big] = -\mathbf{1}_k^\top \bar{v}^{(t)}.$$

The resulting $\bar{u}^{(t)}$ has zero total once the direct terminal contribution has been combined at $t = R$, and thereafter the invariant is zero total on both components. Pulling a zero-total $\bar{u}^{(t)}$ through a unit-row-normalized u-half-step gives a zero-total $\bar{v}^{(t-1)}$. Thus each later centering boundary sees equal totals, so the explicit $\mathcal{C}^*$ correction vanishes in the unit-target proof. If non-unit marginals are introduced, the corresponding normalized conditional kernels must be used in this invariant. □

Lemma 4 (Half-Step Jacobians)**.** *For every refinement step $t \geq 1$,*

$$\frac{\partial u_i^{(t)}}{\partial v_j^{(t-1)}} = -P_{ij}^{(t,t-1)}, \tag{18}$$

$$\frac{\partial v_j^{(t)}}{\partial u_i^{(t)}} = -P_{ij}^{(t,t)}. \tag{19}$$

Proof. Differentiate the masked logsumexp updates. For example,

$$\frac{\partial u_i^{(t)}}{\partial v_j^{(t-1)}} = -\frac{\exp(S_{ij} + v_j^{(t-1)})}{\sum_{k:(i,k)\in\Omega} \exp(S_{ik} + v_k^{(t-1)})} = -\exp(S_{ij} + u_i^{(t)} + v_j^{(t-1)}) = -P_{ij}^{(t,t-1)}.$$

The second identity is identical with rows and columns swapped. □

Proof of Proposition 2.

Proof. By the preceding centering-pullback lemma, the explicit $\mathcal{C}^*$ correction terms vanish at the full-step centering boundaries for the unit-target tail reverse pass. It therefore suffices to write the reverse equations for the raw half-steps, with the centered gauge ledger applied separately when translating mixed-time factors between representatives. The direct terminal transport application contributes $P^{(R,R)} \odot Z$ to $\bar{S}$ together with

$$\bar{v}^{(R)} = g_v, \qquad \bar{u}^{(R)} = g_u - P^{(R,R)}\bar{v}^{(R)}.$$

Now fix a reverse step $t \in \{1, \ldots, R\}$. Pulling $\bar{u}^{(t)}$ back through the u-half-step uses the Jacobian $-P^{(t,t-1)}$:

$$\bar{v}^{(t-1)} = -\big(P^{(t,t-1)}\big)^\top \bar{u}^{(t)}, \qquad \bar{S} \mathrel{-}= P^{(t,t-1)} \odot \big(\bar{u}^{(t)}\mathbf{1}^\top\big).$$

For $t \geq 2$, pulling $\bar{v}^{(t-1)}$ back through the v-half-step uses the Jacobian $-P^{(t-1,t-1)}$:

$$\bar{u}^{(t-1)} = -P^{(t-1,t-1)}\bar{v}^{(t-1)}, \qquad \bar{S} \mathrel{-}= P^{(t-1,t-1)} \odot \big(\mathbf{1}\bar{v}^{(t-1)\top}\big).$$

The last row-half-step produces $\bar{v}^{(0)}$; the training surrogate discards the cotangent on the stopped base state instead of propagating it through the T-step solve. Summing the direct terminal term with the reverse-time half-step contributions yields equation (14). □

Proof of Proposition 3.

Proof. For any two staircase plans and each $(i, j) \in \Omega$,

$$\frac{P_{ij}^{(a,b)}}{P_{ij}^{(2,2)}} = \exp(u_i^{(a)} - u_i^{(2)} + v_j^{(b)} - v_j^{(2)}),$$

which gives the exact reconstruction formula immediately. For the three non-reference factors in the $R = 2$ staircase this gives

$$P_{ij}^{(2,1)} = P_{ij}^{(2,2)}\beta_j, \qquad P_{ij}^{(1,1)} = P_{ij}^{(2,2)}\alpha_i\beta_j, \qquad P_{ij}^{(1,0)} = P_{ij}^{(2,2)}\alpha_i\delta_j.$$

Substituting these three identities into the expanded VJP in equation (15) and factoring out $P_{ij}^{(2,2)}$ gives

$$\bar{S}_{ij} = P_{ij}^{(2,2)}\Big[Z_{ij} - \bar{v}_j^{(2)} - \beta_j\bar{u}_i^{(2)} - \alpha_i\beta_j\bar{v}_j^{(1)} - \alpha_i\delta_j\bar{u}_i^{(1)}\Big].$$

Thus every entry of the direct four-factor adjoint is reproduced exactly from the reference plan and vector modifiers.

For a tile $I \times J$, compute $P_{I,J}^{(2,2)}$ once from $S_{I,J}$, $u_I^{(2)}$, and $v_J^{(2)}$, compute $Z_{I,J}$ from the resident slices of G_I and V_J, and form the bracketed modifier using only the slices of α_I, $\bar{u}_I^{(2)}$, $\alpha_I\bar{u}_I^{(1)}$, β_J, δ_J, $\bar{v}_J^{(2)}$, and $\beta_J\bar{v}_J^{(1)}$. Multiplying the single resident reference tile by this modifier yields the exact score-space VJP tile. The value-gradient term uses the same resident reference plan tile, since $\partial\mathcal{L}/\partial V$ receives $(P_{I,J}^{(2,2)})^\top G_I$. All subsequent propagation to Q and K is linear in the score cotangent tile, so it can be accumulated before the tile is evicted. The stated storage and arithmetic bounds follow by streaming over the $O(LW/B^2)$ active tiles with fixed B, d, W, and fixed tail depth. □

B Dustbin Bridge and Conditional Bias Certificates

B.1 Single-Active-Dustbin Bridge

The current dustbin TPU path does not use a new OT solver. Instead, it augments the unit-target problem. Let the base sequence length be L and let $B = 128$ denote the dustbin block size used in the implementation. Define augmented tensors

$$\widetilde{Q}, \widetilde{K}, \widetilde{V} \in \mathbb{R}^{(L+B)\times d}$$

by appending one learned dustbin token per side together with $B-1$ inactive filler tokens. The masks are augmented analogously: one appended token is active, the rest are inactive fillers. The implementation then applies the same balanced tail-refinement solver to this augmented system, with effective band width

$$\widetilde{W} = \max(W, L).$$

The current reported implementation is exactly the unit-target surrogate on the support $\widetilde{\Omega}_{\mathrm{impl}}$ induced by this widened generic band mask plus inactive-filler masks. A hypothetical sparse-spoke implementation would instead use

$$\widetilde{\Omega}_{\mathrm{spoke}} = \Omega_{\mathrm{band}} \cup \{(i, d_k) : i \in I_q\} \cup \{(d_q, j) : j \in I_k\} \cup \{(d_q, d_k)\},$$

where d_q and d_k are the active appended dustbin row and column; that support would have $|\Omega_{\mathrm{band}}| + O(L)$ active edges. The inactive fillers carry zero mask mass. Because the current reported code path realizes the bridge through the generic widened band mask $\widetilde{W} = \max(W, L)$, its arithmetic should be accounted as $O(L\widetilde{W})$, which becomes dense $O(L^2)$ when $W < L$. We do not claim the current dustbin path has the same long-context complexity as the balanced banded path in that regime.

Proposition 5 (Dustbin Augmentation Bridge)**.** *Let $(\widetilde{Q}, \widetilde{K}, \widetilde{V}, \widetilde{\Omega}_{\mathrm{impl}})$ denote the dustbin-augmented tensors and implemented active support constructed above. Then the current* `dustbin_block` *path is exactly the same unit-target fixed-support tail-refinement surrogate as in Appendix A.2, but applied on the augmented state space.*

Corollary 1 (Exact $R = 2$ Adjoint on the Augmented State Space)**.** *Apply Propositions 2 and 3 to the augmented system. Then the current dustbin-augmented transport path inherits the exact fixed-depth adjoint and the same one-reference-tile schedule on the augmented support.*

This bridge preserves the central finite-step factorization. With one active dustbin token per side, the dustbin marginal capacity is one unit on each side; arbitrary gap capacity would require enlarged dustbin marginals, multiple active dustbin tokens, or a genuinely unbalanced model. Adding one unit-target token on each side does not remove a pre-existing difference between the total query and key target masses.

B.2 A Local Surrogate-Bias Bound

The fixed-depth backward is exact for the surrogate it differentiates. A separate question is how far that surrogate gradient can be from the gradient through the full stopped-base computation. Let

$$z = (u, v), \qquad \Phi_x(z) = z^+,$$

be the centered one-step refinement map at differentiable parameters x, and let $A_x(z)$ denote the transport application map. Write

$$z_T(x) = \Phi_x^T(z_0), \qquad \Psi_x^R(z) = A_x\big(\Phi_x^R(z)\big).$$

The transport application is vector-valued, so we first compare its output Jacobians. Write

$$J_{\mathrm{full}}^{T,R}(x) = D_x\big[\Psi_x^R(z_T(x))\big],$$

while the stop-gradient surrogate uses

$$J_{\mathrm{surr}}^{T,R}(x) = D_x \Psi_x^R(z)\big|_{z=z_T(x)},$$

where the latter derivative holds z fixed. Below, D denotes a Jacobian with output coordinates indexing rows. For scalar objectives, gradients are column vectors, and $\partial_x F$ denotes the partial column gradient with the state held fixed.

Theorem 1 (Local Surrogate-Bias Bound)**.** *Assume that the initial state z_0 is independent of x, fixed support, C^1 dependence of Φ_x and A_x on (x, z), and constants $\rho \in [0, 1)$, $L_\Phi > 0$, and $L_A > 0$ such that*

$$\|D_z\Phi_x(z)\| \le \rho, \qquad \|D_x\Phi_x(z)\| \le L_\Phi, \qquad \|D_zA_x(z)\| \le L_A$$

on a forward-invariant neighborhood, using compatible operator norms. Then

$$J^{T,R}_{\text{full}}(x) - J^{T,R}_{\text{surr}}(x) = D_z\Psi^R_x(z_T(x))\, D_xz_T(x),$$

and

$$\left\|J^{T,R}_{\text{full}}(x) - J^{T,R}_{\text{surr}}(x)\right\| \le \frac{L_AL_\Phi}{1-\rho}\,\rho^R.$$

For a fixed differentiable scalar loss ℓ, let

$$F_R(x, z) = \ell\big(\Psi^R_x(z)\big)\,.$$

If additionally $\|D_z(\ell \circ A_x)(z)\| \le L_{\ell\circ A}$ on the same neighborhood, the full and stopped-base column gradients satisfy

$$g^{T,R}_{\text{full}} - g^{T,R}_{\text{surr}} = \big(D_xz_T\big)^\top \nabla_z F_R(x, z_T(x)),$$
$$\left\|g^{T,R}_{\text{full}} - g^{T,R}_{\text{surr}}\right\| \le \frac{L_{\ell\circ A}L_\Phi}{1-\rho}\,\rho^R.$$

where $g^{T,R}_{\text{full}} = \nabla_x[F_R(x, z_T(x))]$ and $g^{T,R}_{\text{surr}} = \partial_xF_R(x, z)|_{z=z_T(x)}$.

Under these fixed-support local-contraction assumptions, the stop-gradient bias decays geometrically with tail depth R. We next give the finite-instance certificate used by the validation harness.

Proposition 6 (Projective Sinkhorn Contraction Certificate)**.** *Consider the two-half-step map restricted to a fully active rectangular block $I \times J$ of the fixed or dustbin-augmented support, and let*

$$\mathsf{K}_{ij} = \exp(S_{ij})$$

be the positive kernel on that block, using the scaled score convention from Appendix A.1. For positive column scalings c, c', define the two-half-step Sinkhorn scaling map

$$\mathcal{T}(c) = b \oslash \left[\mathsf{K}^\top (a \oslash (\mathsf{K}c))\right],$$

with positive marginals a, b; in our normalized update $a = b = \mathbf{1}$. Let d_H denote Hilbert's projective metric on the positive cone and let $\Delta(M)$ be the projective diameter of a positive linear map M. Then

$$d_H(\mathcal{T}(c), \mathcal{T}(c')) \le \rho_H\, d_H(c, c'), \qquad \rho_H = \tanh\left(\frac{\Delta(\mathsf{K})}{4}\right)\tanh\left(\frac{\Delta(\mathsf{K}^\top)}{4}\right) < 1.$$

Equivalently, in log scalings, with $\Phi_{\text{col}}(v) = \log \mathcal{T}(e^v)$ modulo additive gauge, the centered half-step pair is a contraction in oscillation seminorm:

$$\operatorname{osc}(\Phi_{\text{col}}(v) - \Phi_{\text{col}}(v')) \le \rho_H\, \operatorname{osc}(v - v').$$

If $\Omega_S = \max_{(i,j)} S_{ij} - \min_{(i,j)} S_{ij}$ on the active block, then a computable sufficient bound is

$$\rho_H \le \tanh^2\left(\frac{\Omega_S}{2}\right) < 1.$$

Equivalently, for unscaled logits A with $S = A/\varepsilon$, this reads

$$\rho_H \le \tanh^2\left(\frac{\Omega_A}{2\varepsilon}\right).$$

This proposition is a support-local certificate, not a claim about masked zeros or the entire coupled masked solve. The Birkhoff–Hopf argument [Birkhoff, 1957, Bushell, 1973] applies on strictly positive rectangular blocks after masked rows/columns and inactive dustbin fillers have been excluded; excluding individual masked entries alone does not produce a positive matrix. The dustbin bridge preserves the statement on any fully active augmented block, while the inactive filler tokens are outside the cone on which the contraction is asserted. The measured block coefficients do not by themselves verify the full-state operator-norm assumption of Theorem 1.

Proposition 7 (A Posteriori Bias Certificate)**.** *Let $F_R(x, z)$ denote the scalar tail objective obtained by applying the terminal loss to $A_x(\Phi_x^R(z))$, and define the measured base-state cotangent*

$$\eta_R(x) = \nabla_z F_R(x, z)\big|_{z=z_T(x)}.$$

Then the full-gradient remainder omitted by the stopped-base surrogate is exactly

$$\nabla_x F_R(x, z_T(x)) - \partial_x F_R(x, z)\big|_{z=z_T(x)} = D_x z_T(x)^\top \eta_R(x).$$

Consequently, for any compatible norm,

$$\|\Delta g_R\| \le \|D_x z_T(x)^\top\| \, \|\eta_R(x)\|.$$

This certificate is operational: η_R is the base cotangent produced by the same fixed-depth tail reverse pass before it is discarded by the stop-gradient rule, and $D_x z_T(x)^\top \eta_R$ is a single VJP through the base Sinkhorn solve.

Corollary 2 (Certified Tail-Depth Selection)**.** *Fix a tolerance $\tau > 0$ and a maximum tail depth $R_{\max}$. For each $R \le R_{\max}$, let*

$$c_R(x) = D_x z_T(x)^\top \eta_R(x).$$

Let $C(R)$ be any nondecreasing implementation cost over the candidate depths $\{0, \dots, R_{\max}\}$. If the implementation chooses the first R such that $\|c_R(x)\| \le \tau$ and returns the stopped-base gradient at that depth, then the returned gradient differs from full BPTT through the same T-step base solve by at most τ. Moreover, this first feasible R solves the discrete certified optimization problem

$$\min_{0 \le R \le R_{\max}} C(R) \quad \textit{subject to} \quad \|c_R(x)\| \le \tau.$$

If an upper bound $C_T(x) \ge \|D_x z_T(x)^\top\|$ is available, the same guarantee and optimality statement hold with the sufficient constraint $C_T(x)\|\eta_R(x)\| \le \tau$.

Thus one may choose the least-cost tail depth satisfying the certificate, use a calibrated fixed depth such as $R = 2$, or periodically evaluate the certificate VJP to verify that the omitted stopped-base term remains below the chosen tolerance.

Proposition 8 (Transport-Orbit Calculus)**.** *Consider any finite collection of transport plans $P^{(a,b)}$ generated from the same masked score tensor S and the same active support, including the plans used by the $R = 2$ staircase, the T stopped base reverse pass, the R differentiable tail reverse pass, and the certificate VJP $D_x z_T^\top \eta_R$. This fixed-score condition is essential: under an ε-scaling continuation schedule, the orbit statement applies stagewise to plans using the same effective score $S^{(s)} = A/\varepsilon_s$, not across different temperature stages. Fix any reference plan $P^\star = P^{(a_\star, b_\star)}$ in this collection and define*

$$r^{(a)} = \exp(u^{(a)} - u^{(a_\star)}), \qquad s^{(b)} = \exp(v^{(b)} - v^{(b_\star)}).$$

Then every plan in the collection lies in the row/column scaling orbit of $P^\star$:

$$P^{(a,b)} = \operatorname{diag}(r^{(a)}) \, P^\star \, \operatorname{diag}(s^{(b)}).$$

Equivalently, for any finite sum of plan-weighted score cotangent terms

$$\mathcal{G} = \sum_{\ell=1}^{m} P^{(a_\ell, b_\ell)} \odot H_\ell,$$

one has the single-reference representation

$$\mathcal{G} = P^\star \odot \sum_{\ell=1}^{m} \big(r^{(a_\ell)} (s^{(b_\ell)})^\top\big) \odot H_\ell.$$

Consequently, for fixed T, R, band width W, and head dimension d, the stopped gradient, the certified omitted-gradient VJP, and the certificate test at that candidate tail depth can all be evaluated blockwise with $O((T + R)LW)$ arithmetic, $O(Ld)$ input storage, and $O((T + R)L)$ vector state, while materializing only one transport tile at a time. The same single-orbit streaming schedule therefore covers the stopped gradient and both certificate computations within a fixed-score stage. Searching multiple candidate depths additionally incurs the costs of the candidates evaluated; the certified selection result minimizes the chosen depth's cost $C(R)$ and does not assert that the search overhead is free.

B.3 Proofs

Proof of Proposition 5.

Proof. After constructing the augmented tensors and masks, the implementation calls the same unit-target Sinkhorn forward and backward path as in Appendix A.2. Because only the first appended token in each dustbin block is active, every transport entry touching the remaining fillers is identically zero. Because the effective band width is $\widetilde{W} = \max(W, L)$, every active base token can reach the active dustbin token on the opposite side, and conversely the active dustbin query can reach every active base key. Thus the implemented path is exactly the unit-target fixed-support surrogate on an enlarged state space. The proposition is an algebraic code-path statement; the cost of the current widened-band realization is governed by $\widetilde{W}$, as stated in Appendix B.1. □

Proof of Corollary 1.

Proof. Apply Propositions 2 and 3 to the augmented tensors and support from Proposition 5. No new algebra is introduced after the augmentation itself. □

Proof of Theorem 1.

Proof. By the chain rule,

$$D_x\big[\Psi_x^R(z_T(x))\big] = D_x\Psi_x^R(z)\big|_{z=z_T(x)} + D_z\Psi_x^R(z_T(x))\, D_x z_T(x),$$

which gives the exact bias identity after subtracting the partial derivative term.

Next,

$$D_z\Psi_x^R(z_T(x)) = D_z A_x(z_{T+R}(x)) D_z\Phi_x(z_{T+R-1}(x)) \cdots D_z\Phi_x(z_T(x)),$$

with the identity map as the empty product for $R = 0$. Therefore

$$\|D_z\Psi_x^R(z_T(x))\| \le L_A \rho^R.$$

Differentiate the base recursion $z_{t+1}(x) = \Phi_x(z_t(x))$:

$$D_x z_{t+1}(x) = D_x\Phi_x(z_t(x)) + D_z\Phi_x(z_t(x))\, D_x z_t(x).$$

Because z_0 is independent of x, repeated application of the norm bounds yields

$$\|D_x z_T(x)\| \le L_\Phi \sum_{j=0}^{T-1} \rho^j \le \frac{L_\Phi}{1-\rho}.$$

Combining the two estimates with the exact bias identity gives

$$\big\|J_{\text{full}}^{T,R}(x) - J_{\text{surr}}^{T,R}(x)\big\| \le \frac{L_A L_\Phi}{1-\rho}\, \rho^R.$$

For the scalar objective, the column-gradient chain rule gives the transposed identity in the theorem. Replacing the terminal map A_x by $\ell \circ A_x$ in the preceding derivative estimate bounds $\|\nabla_z F_R\|$ by $L_{\ell\circ A}\rho^R$. Combining that bound with the same estimate for $\|D_x z_T\|$ proves the scalar-gradient inequality. □

Proof of Proposition 6.

Proof. Hilbert's projective metric on the positive cone is

$$d_H(x, y) = \log \frac{\max_i x_i/y_i}{\min_i x_i/y_i}.$$

Positive diagonal scaling and coordinatewise inversion preserve this metric: diagonal scaling cancels in the coordinate ratios, and inversion swaps the maximum and minimum ratios. By the Birkhoff–Hopf theorem, any positive linear map M with finite projective diameter $\Delta(M)$ satisfies

$$d_H(Mx, My) \le \tanh\left(\frac{\Delta(M)}{4}\right) d_H(x, y).$$

Decompose the two-half-step map as

$$c \mapsto \mathsf{K}c \mapsto a \oslash (\mathsf{K}c) \mapsto \mathsf{K}^\top (a \oslash (\mathsf{K}c)) \mapsto b \oslash \mathsf{K}^\top (a \oslash (\mathsf{K}c)).$$

The two linear maps contribute Birkhoff–Hopf contraction factors $\tanh(\Delta(\mathsf{K})/4)$ and $\tanh(\Delta(\mathsf{K}^\top)/4)$, while the marginal scalings and inversions are projective isometries. Therefore

$$d_H(\mathcal{T}(c), \mathcal{T}(c')) \le \tanh\left(\frac{\Delta(\mathsf{K})}{4}\right) \tanh\left(\frac{\Delta(\mathsf{K}^\top)}{4}\right) d_H(c, c').$$

Strict positivity on the active block gives finite projective diameters, hence $\rho_H < 1$.

For log scalings,

$$d_H(e^v, e^{v'}) = \operatorname{osc}(v - v'), \qquad \operatorname{osc}(w) = \max_i w_i - \min_i w_i,$$

and additive centering does not change oscillation. This gives the log-potential contraction.

Finally, for a positive matrix $\mathsf{K}_{ij} = \exp(S_{ij})$,

$$\Delta(\mathsf{K}) = \log \max_{i,i',j,j'} \frac{\mathsf{K}_{ij}\mathsf{K}_{i'j'}}{\mathsf{K}_{ij'}\mathsf{K}_{i'j}} = \max_{i,i',j,j'} \left(S_{ij} + S_{i'j'} - S_{ij'} - S_{i'j}\right).$$

If $\Omega_S = \max S - \min S$ on the active block, the last expression is at most $2\Omega_S$. The same bound holds for $\mathsf{K}^\top$. Substituting into the Birkhoff–Hopf coefficient yields

$$\rho_H \le \tanh^2\left(\frac{\Omega_S}{2}\right),$$

and the unscaled-logit form follows from $S = A/\varepsilon$. □

Proof of Proposition 7.

Proof. By definition,

$$g_{\text{full}}(x) = \nabla_x F_R(x, z_T(x)), \qquad g_{\text{surr}}(x) = \partial_x F_R(x, z)\big|_{z=z_T(x)}.$$

Applying the chain rule to the first expression gives

$$\nabla_x F_R(x, z_T(x)) = \partial_x F_R(x, z)\big|_{z=z_T(x)} + D_x z_T(x)^\top \nabla_z F_R(x, z)\big|_{z=z_T(x)}.$$

Subtracting the stopped-base partial derivative gives the identity with $\eta_R = \nabla_z F_R(x, z_T(x))$. The norm bound follows immediately from submultiplicativity for the chosen compatible operator norm. □

Proof of Corollary 2.

Proof. Proposition 7 gives the exact identity $\Delta g_R = c_R(x)$. Therefore $\|c_R(x)\| \le \tau$ implies $\|\Delta g_R\| \le \tau$. If only an operator-norm bound $C_T(x) \ge \|D_x z_T(x)^\top\|$ is used, the bound in Proposition 7 gives

$$\|\Delta g_R\| \le \|D_x z_T(x)^\top\| \, \|\eta_R(x)\| \le C_T(x)\|\eta_R(x)\| \le \tau.$$

For the optimization claim, let $R^\star$ be the first feasible depth. Every smaller depth is infeasible by definition of $R^\star$, and every larger feasible depth has cost at least $C(R^\star)$ because C is nondecreasing. Hence $R^\star$ minimizes $C(R)$ over the certified feasible set. □

Proof of Proposition 8.

Proof. All plans in the collection use the same masked score tensor S and the same active support. If an ε-scaling schedule is used, this proof applies separately inside each fixed-temperature stage after replacing S by $S^{(s)} = A/\varepsilon_s$; plans from different stages are not generally in one common row/column scaling orbit. For any two finite dual pairs and each (i, j) on that active support,

$$\frac{P_{ij}^{(a,b)}}{P_{ij}^{(a_\star,b_\star)}} = \exp(u_i^{(a)} - u_i^{(a_\star)} + v_j^{(b)} - v_j^{(b_\star)}),$$

which gives the orbit formula entrywise. Multiplying this identity by H_ℓ and summing over ℓ gives the single-reference representation for $\mathcal{G}$. The reverse recurrences in Propositions 2 and 7 require only products of these plans with row or column cotangent vectors and score-space outer factors, all of which are instances of this representation. Each such product can therefore be evaluated by streaming one reference plan tile and applying the appropriate row and column multipliers. With fixed T, R, W, and d, each half-step contributes $O(LW)$ arithmetic and $O(L)$ vector state, while the input activations require $O(Ld)$ storage. These bounds are for one candidate tail depth; the costs sum over the depths evaluated by a search. □

C Complete Empirical Results and Protocols

C.1 Exactness Validation of the Masked Surrogate

On synthetic masked problems for which exact autodiff through the same centered surrogate remains tractable, we compare the optimized $R = 2$ kernel against a pure-JAX reference. Table 4 reports representative maximum absolute gradient deviations under the default validation setting: float32 arithmetic, $\varepsilon = 1$, no ε-scaling continuation, band width 256, $R = 2$, centered potentials, and masked inactive entries zeroed.

Length L	$\max \lvert\Delta Q\rvert$	$\max \lvert\Delta K\rvert$	$\max \lvert\Delta V\rvert$	Max rel. L_2
512	1.05×10^{-5}	1.73×10^{-6}	1.06×10^{-9}	5.78×10^{-2}
1024	5.70×10^{-6}	8.13×10^{-7}	2.55×10^{-10}	5.76×10^{-2}
2048	5.50×10^{-6}	9.17×10^{-7}	2.29×10^{-10}	5.05×10^{-2}

Table 4: Gradient deviations between the optimized masked $R = 2$ kernel and exact autodiff through the same centered surrogate. The final column is the worst relative L_2 error among $(\partial Q, \partial K, \partial V)$; the corresponding worst absolute L_2 errors are 6.59×10^{-5}, 4.11×10^{-5}, and 2.62×10^{-5}.

The table is primarily a kernel-equivalence check, not a full numerical-analysis study. The relative L_2 errors are largest for the small-norm Q gradients, whose reference L_2 norms are 1.14×10^{-3}, 7.14×10^{-4}, and 5.18×10^{-4} in the three rows; the absolute L_2 errors remain below 6.6×10^{-5} and the output relative L_2 errors are below 1.8×10^{-7}.

Additional forward-consistency checks run through lengths 512, 1024, 2048, 4096, 8192, and 16384, with extended spot checks at 32768, 65536, and 131072. We treat the larger checks as forward-consistency evidence rather than as exact-gradient checks.

To calibrate the local surrogate-bias discussion, we also compared tail-refinement gradients against full BPTT through the stopped-base solve on the same synthetic masked setup at $L = 512$, $d = 8$, $T = 15$, $W = 256$, and $\varepsilon = 1$. Over seeds $0, 1, 2$, the worst-case gradient discrepancy $\max\{|\Delta Q|, |\Delta K|, |\Delta V|\}$ was 1.49×10^{-4} for $R = 0$, 1.63×10^{-5} for $R = 1$, 1.42×10^{-5} for $R = 2$, and 1.42×10^{-5} for $R = 4$. In this local exact-gradient calibration, the main empirical drop therefore occurs between zero tail refinement and the first one or two refinement steps, while $R = 2$ and $R = 4$ are already effectively tied at the scale of the masked validation harness.

Table 5 reports the a posteriori certificate from Proposition 7 on a smaller full-BPTT setting where the base-solve VJP can be checked directly ($L = 128$, $d = 8$, $T = 15$, $W = 128$, seeds $0, 1, 2$). The certificate reconstructs the full-gradient remainder to numerical precision and shows rapid decay of the measured base-state cotangent.

Tail depth R	$\max \lVert\eta_R\rVert_2$	$\max \lvert\Delta g_R\rvert$	certificate residual
0	5.89×10^{-3}	6.29×10^{-4}	2.47×10^{-10}
1	5.40×10^{-4}	5.93×10^{-5}	3.64×10^{-11}
2	5.47×10^{-5}	5.33×10^{-6}	2.84×10^{-11}
4	9.03×10^{-7}	9.99×10^{-8}	2.74×10^{-11}

Table 5: A posteriori surrogate-bias certificate. The residual is the maximum absolute difference between the actual full-vs-stopped gradient gap and the certified base-solve VJP $D_x z_T^\top \eta_R$.

Using the certified selector from Corollary 2 with max-absolute tolerance 10^{-5} selects $R = 2$ on all three seeds in this calibration. Since the per-depth tail cost is monotone in R, this is the cheapest certified depth among the tested candidates and gives a certificate-level justification for using $R = 2$ as the reported TPU depth.

We also validate the transport-orbit calculus from Proposition 8 directly over all $2(T+R)$ base and tail half-step plans at $T = 15, R = 2, L = 128$. Reconstructing every plan from the final reference plan gives maximum log-domain reconstruction error 1.91×10^{-6}. Appendix C.9 reports two additional diagnostics. The controlled direct-four-plan score-adjoint diagnostic reduces logical active plan-factor storage by exactly $4\times$ and gives 1.29–$2.20\times$ post-compilation CPU speedups with maximum score-cotangent discrepancy at most 4.77×10^{-7}; it is still a dense-JAX CPU materialization check, not a production TPU systems benchmark. The same appendix also reports projective contraction certificates computed from the promoted Pfam checkpoint.

C.2 Literal Same-Device V5E1 Production Backward Matrix

We ran four modes on one managed TPU V5E1 device—not a v6e-8 slice—at $B = 1, H = 1, d = 64, T = 15, R = 2, W = 1024$, with $\varepsilon = 1$, seed 0, float32 inputs, and explicit `matmul_precision=float32`, using JAX/JAXlib 0.6.2 and libtpu 0.0.17: (A) naive four-resident-plan materialization, (B) direct-four tile-local recomputation, (C) the exact one-reference factorization, and (D) exact autodiff. The matrix covers $L \in \{2048, 8192, 16384\}$ with one fresh process per mode and length, five warmups, and twenty synchronized measurements. All twelve publication processes succeeded. Across validated comparisons, maximum loss, output, and $Q/K/V$-gradient discrepancies are 1.24×10^{-5}, 4.47×10^{-8}, and 2.85×10^{-5}.

Table A: saved-state backward-only latency (median ms). The saved-state backward-only benchmark times the synchronized production consumer after its declared saved representation has already been created. Exact autodiff is marked not applicable because it exposes compiler-managed residuals rather than the same public saved-state consumer contract.

L	Naive four-resident	Direct four/recompute	One-reference	Exact autodiff
2048	0.81	2.56	9.72	–
8192	3.00	10.99	47.23	–
16384	6.10	21.74	96.48	–

Table B: complete-step latency and compiler-total memory. The one-executable complete-step benchmark includes forward computation, state materialization and retention, and arbitrary-cotangent $Q/K/V$ backward in every row.

L	Mode	Complete ms	Compiler total MiB	Saved payload MiB	Dynamic XProf peak
2048	Naive four-resident	30.62	70.3	64.0	–
2048	Direct four/recompute	29.70	7.0	0.0	–
2048	One-reference	36.90	7.0	0.0	–
2048	Exact autodiff	37.25	248.3	0.0	–
8192	Naive four-resident	125.88	290.4	272.0	–
8192	Direct four/recompute	128.04	18.6	0.1	–
8192	One-reference	164.12	18.6	0.1	–
8192	Exact autodiff	188.32	1428.9	0.0	–
16384	Naive four-resident	257.44	578.5	544.0	–
16384	Direct four/recompute	260.11	34.7	0.1	–
16384	One-reference	334.91	34.7	0.1	–
16384	Exact autodiff	393.22	3204.1	0.0	–

The latency ordering depends on scope and length. In Table A, naive four-resident wins at $L = 2048/8192/16384$ with medians $0.81/3.00/6.10$ ms; direct-four records $2.56/10.99/21.74$ ms and one-reference $9.72/47.23/96.48$ ms. In Table B, direct-four wins the $L = 2048$ complete step at 29.70 ms, while naive wins at $L = 8192$ and 16384 with 125.88 and 257.44 ms; direct-four is close at 128.04 and 260.11 ms, and one-reference records 164.12 and 334.91 ms. Thus the current one-reference lowering is not the latency winner, and no universal speed claim is authorized.

The representation result is nevertheless exact and directly visible in the complete-step state. The naive HLO audit verifies four distinct materialization and consumption markers, four retained plan outputs, numerically distinct plans, a retention barrier, synchronized outputs, and no dead-code elimination, all in one compiled executable. Its explicit float32 plan payload is 64, 272, and 544 MiB. Direct-four and one-reference retain only 0.015625, 0.0625, and 0.125 MiB of explicit vector state and no full plan array; one-reference obtains its backward factors from the exact four-to-one staircase-plan factorization.

At $L = 16384$, the complete-step medians correspond to 63.6, 63.0, 48.9, and 41.7 thousand positions/s for literal retention, direct recomputation, one-reference, and exact autodiff, respectively. These throughputs include forward computation and the arbitrary-cotangent backward pass, rather than only the saved-state consumer.

The compiler-total column is the executable's arguments plus outputs plus temporaries minus aliases. It already includes the explicit saved payload and must not be added to that payload again. At $L = 16384$, compiler total is 578.5 MiB for naive, 34.7 MiB for direct-four, 34.7 MiB for one-reference, and 3204.1 MiB for exact autodiff. A separately attributable dynamic XProf peak was unavailable for every row; "–" means unavailable, not zero, and compiler-total analysis is not relabelled as dynamic peak HBM. The V2 contract intentionally stops at the three reviewer-requested lengths, so earlier implementation-specific feasibility probes are not reused as V2 timing or memory evidence.

C.3 Exact Complete Backward at Extreme Context

We also evaluate complete forward-plus-arbitrary-cotangent $Q/K/V$ VJPs on all eight devices of a TPU v6e-8 host. One process uses data-parallel `pmap`, with one distinct full sequence on each device and no sequence sharding. Inputs are BF16, with $H = 1, d = 64, W = 1024, T = 15, R = 2$, seed 0, and float32 matmul precision. Table 6 reports one synchronized warmup followed by three synchronized measurements per case.

Context/device	Exact schedule	Complete VJP (s)	Compiler total (MiB)
131072	One-reference	2.993	354.684
131072	Direct recomputation	2.709	354.622
262144	One-reference	6.020	835.715
262144	Direct recomputation	5.415	835.528

Table 6: Exact stopped-base $R = 2$ complete VJPs on eight independent full sequences. Compiler totals are static executable estimates in the reported `pmap` accounting scope; that byte scope is not established as either per-device or aggregate, and it is not dynamic peak HBM.

The explicit value-and-VJP interface avoids transformation-generated scan tapes while preserving the surrogate and its gradients. The earlier lowering reached a scoped-VMEM compilation boundary at 131072 for both low-state schedules; the explicit windowed route completes that point and 262144. At 16384, its output and $dQ/dK/dV$ are byte-identical to the frozen earlier witnesses for one-reference, direct recomputation, and literal retention. This checks the implementation change at the tractable witness point rather than claiming an exact-autodiff check at 262144.

A separate allocation with the same device class, runtime configuration, and exact 262144-position inputs reports 9347.184 MiB for literal four-plan retention, compared with the one-reference total of 835.715 MiB. One-reference is therefore $11.18\times$ smaller in this compiler accounting. That historical run used two warmups and five measurements rather than one and three, so this is a descriptive state comparison across allocations and protocols, not a same-run speedup. Native dense FlashAttention and same-support Local SplashAttention also complete at 262144: the former changes support and the latter computes sparse softmax, so neither is an equal-operator ranking. Dynamic peak HBM remains unavailable, and 262144 is a tested context rather than a maximum feasible length.

C.4 Official FlashSinkhorn Dense Forward Comparisons

Table 7 compares a deliberately separate dense JAX compatible-marginal reference with the pinned official `flash-sinkhorn==0.3.3.post1` package on the same physical Tesla T4. It is not a

N	Median (ms)		Official/Dense ref.	Throughput (M entries/s)		Max $\lvert\Delta\rvert$
	Dense JAX ref.	Official	time ratio	Dense JAX ref.	Official	timed output
512	1.248	4.503	3.607	209.915	59.879	8.73×10^{-10}
1024	1.995	4.587	2.299	525.616	226.853	4.07×10^{-10}
2048	5.412	11.536	2.131	750.991	363.694	2.04×10^{-10}
4096	47.747	39.631	0.830	383.444	422.200	8.73×10^{-11}

Table 7: Same-input dense compatible-marginal forward solver-plus-transport benchmark on one Tesla T4. Timings exclude compilation, correctness checks, and input allocation. All correctness gates pass; JAX is a dense reference, not our blockwise Pallas kernel. This table does not measure backward, arbitrary masks, Pfam training, or end-to-end model runtime.

benchmark of our submitted blockwise Pallas kernel. For this earlier table, an A100 was requested but unavailable, so it is explicitly a T4 fallback result. Each case uses the same dense square inputs, compatible unit-mass marginals, float32 arithmetic, head and value dimension 64, and 15 solver iterations. Timings are CUDA-synchronized over 20 repetitions after 5 warmups and exclude compilation, correctness gates, and input allocation.

The dense JAX reference records lower medians at $N = 512$, 1024, and 2048; official FlashSinkhorn records the lower median at $N = 4096$. The dense-reference $N = 4096$ samples are nonstationary: the first three measured repetitions are approximately 21.7–23.0 ms, whereas most later repetitions are approximately 43.1–48.3 ms. We therefore retain the recorded mean, median, and p95 in the evidence artifact and do not claim a smooth crossover. Across all four cases, the maximum timed-output absolute discrepancy is at most 8.73×10^{-10}, and the maximum dense-reference-versus-official discrepancy in the 64×64 small-reference gate is 1.12×10^{-8}.

A separate actual-kernel diagnostic binds every isolated child to the same physical A100 UUID and uses byte-identical float32-storage $q/k/v$ inputs, dimension 64, and 17 Sinkhorn updates. These controls fix device and input identity only. To construct the supported common forward surface, we expand our band to $W = L$. At $L = 512/1024/2048/4096$, official Triton medians are $3.717/3.773/5.727/15.549$ ms, while our TPU-oriented JAX/Pallas program lowered non-natively through CUDA records $34.014/114.653/411.212/1700.474$ ms. This is not an equivalent-native or method-speed comparison: the execution paths differ, $W = L$ removes the submitted $O(LW)$ regime, the arithmetic and timing scopes are not strict matches, and Pallas normalization outside timing favors our path. An independent post-hoc replay of the immutable arrays gives relative-L_2 discrepancies of 4.10×10^{-4}–4.44×10^{-4}. Official FlashSinkhorn also succeeds with finite output and without OOM through $L = 50000$. The fixed-band GPU complete-step path instead stops at an unsupported JAX Pallas `slice` lowering, not OOM. The fixed-band forward and exact $Q/K/V$ backward are measured on V5E1 through $L = 16384$; Section C.3 separately reports the explicit complete-VJP route on v6e-8 through $L = 262144$.

The memory telemetry is deliberately not reduced to a cross-backend ratio. JAX reports a process/device high-watermark that can include bounded correctness and timed execution, while PyTorch reports allocator allocated/reserved high-watermarks after an allocator reset; the official interface also exposes no portable compiler-memory analysis object. These scopes are not directly comparable. More broadly, the validated official interface supplies neither an equivalent arbitrary band mask nor a benchmarked backward contract. Splitting the band into independent dense problems would change the shared normalization, while finite penalties outside the band would replace the hard support. The requested matched-Pfam FlashSinkhorn comparison therefore remains unavailable. The T4 table supports only its separate dense-reference surface, while the A100 run supports only the forced-dense actual-kernel forward diagnostic. Neither supports a matched backward, masked-Pfam, trained-model, cross-backend-memory, or multi-device claim.

C.5 Matched Three-Seed Pfam Reconstruction Screen

We completed a canonical six-row comparison between Sinkhorn OT and trained JAX/Pallas FlashAttention. Each method uses the same real-Pfam examples, trainable projection tree, initialization policy, reconstruction-only objective, optimizer policy, prespecified seed, exactly 100 updates, and an evaluation of exactly $5{,}000$ validation examples. Training uses the native BF16 TPU kernels for each

method; canonical FlashAttention quality is evaluated with a dense float32 attention reference on the trained projections. The matched screen learns query/key projections and stops gradients through the value projection, keeping its reconstruction target fixed. The comparison therefore tests matched short-horizon reconstruction quality, not isolated kernel latency or fused-BF16 evaluator parity.

A. Reconstruction quality

Method	Seed	MSE	Example-bootstrap interval
Sinkhorn	0	0.054595	[0.053835, 0.055395]
Sinkhorn	1	0.052788	[0.051996, 0.053611]
Sinkhorn	2†	0.052374	[0.051630, 0.053165]
FlashAttention	0	0.052842	[0.052100, 0.053639]
FlashAttention	1	0.050680	[0.049959, 0.051433]
FlashAttention	2	0.050782	[0.050056, 0.051538]

B. Recorded execution telemetry

Method	Seed	Elapsed (s)	Updates/s
Sinkhorn	0	2226.2	0.04
Sinkhorn	1	2202.0	0.05
Sinkhorn	2†	n/a†	n/a†
FlashAttention	0	2163.5	0.05
FlashAttention	1	2151.1	0.05
FlashAttention	2	2415.0	0.04

Table 8: Matched real-Pfam reconstruction after exactly 100 updates for three prespecified seeds and exactly 5000 validation examples per run. Lower MSE is better; brackets give per-example bootstrap intervals. FlashAttention denotes trained JAX/Pallas FlashAttention, evaluated with the canonical dense float32 reference on trained projections, not evidence of deployed-BF16 fused-kernel parity. All six rows share the single-device V5E1 hardware class and topology; the same physical TPU is not proven. Every scientific row binds checkpoints at updates $25/50/75/100$. Five rows are ordinary completed, verified, cleaned zero-exit chains. The † Sinkhorn seed-2 result is a content-addressed scientific recovery after a terminal control-plane failure and nonzero systemd exit; its elapsed/rate cells are suppressed and it is ineligible for lifecycle or systems claims. Elapsed time covers the whole invocation, including compilation, data handling, diagnostics, and checkpointing; rates are computed as $100/\text{elapsed}$ and displayed rounded. All six rows' recorded telemetry is bound across the aggregate and parent receipt. Paired differences appear in Table 9. This is short-horizon reconstruction-only validation evidence, not long-horizon, alignment, independent-test, isolated-latency, model-throughput, HBM, or equivalent-hardware speedup evidence.

Sinkhorn reconstruction MSE is 0.0545948, 0.0527878, and 0.0523736 for seeds 0, 1, and 2; trained FlashAttention MSE is 0.0528423, 0.0506797, and 0.0507820. The paired Sinkhorn-minus-FlashAttention differences are $+0.0017525$, $+0.0021081$, and $+0.0015915$, with mean $+0.0018174$ and sample standard deviation 0.0002643. All three observed pairs favor FlashAttention on this 100-update reconstruction screen. With three fixed seeds, the across-seed dispersion and resampled interval in Table 9 are descriptive limited-seed summaries, not population uncertainty or evidence of long-horizon convergence.

All rows share the same single-device V5E1 accelerator class and topology, but physical TPU identity was not held constant. Whole-invocation telemetry includes compilation, data handling, diagnostics, and checkpointing, so it is not an isolated-latency, wall-clock-matched, equivalent-hardware, or method-level speed comparison. The recovered Sinkhorn seed-2 row contributes science only and is excluded from timing, throughput, systems, and ordinary-success claims. This screen optimizes reconstruction only and supplies no biological-alignment-superiority claim.

Statistic	Sinkhorn OT	Trained FlashAttention	Paired Δ (Sinkhorn$-$FA)
Seed 0	0.054595	0.052842	+0.001753
Seed 1	0.052788	0.050680	+0.002108
Seed 2	0.052374†	0.050782	+0.001592
Mean	0.053252	0.051435	+0.001817
Sample s.d.	0.001181	0.001220	0.000264
Minimum	0.052374	0.050680	+0.001592
Maximum	0.054595	0.052842	+0.002108
Descriptive seed interval	[0.052374, 0.054595]	[0.050680, 0.052842]	[+0.001592, +0.002108]

Table 9: Observed real-Pfam validation reconstruction across the three prespecified 100-update seeds, with exactly 5000 validation examples per run. The intervals resample only these three fixed seeds and are descriptive, not population confidence intervals. Paired differences use Sinkhorn minus trained JAX/Pallas FlashAttention; negative values would favor Sinkhorn on reconstruction MSE. All six scientific rows bind checkpoints at updates $25/50/75/100$ and share the single-device V5E1 hardware class and topology, without proven physical-TPU identity. FlashAttention quality uses the canonical dense float32 reference on trained projections, not deployed-BF16 fused-kernel parity. Table 8 reports all execution telemetry once: whole-invocation elapsed time includes compilation, data handling, diagnostics, and checkpointing, and displayed update rates derive from $100/\text{elapsed}$. This is not isolated kernel timing, model throughput, or an equivalent-hardware speedup. Five rows are ordinary completed, verified, cleaned zero-exit chains. The † Sinkhorn seed-2 row is a content-addressed scientific recovery after a terminal control-plane failure and nonzero systemd exit; its telemetry is suppressed and it is ineligible for lifecycle or systems claims. These results do not establish long-horizon stability, alignment or biological superiority, independent-test performance, latency, or HBM.

C.6 Pfam Topology Transfer on TPU v6e-8

As a cross-topology training transfer, all four short-screen configurations complete end-to-end on TPU v6e-8:

`k2_eps1`, `k4_eps1`, `muot_dustbin_mono`, `muot_dustbin_hybrid`.

Each uses 50,000 training pairs, $T = 15$ base iterations, band width 1024, and global batch size 64. Table 10 reports the step-50 screen snapshot.

Config	Step 50 Loss	Step 50 CE	Step 50 Mono	Throughput (ex/s)
`k2_eps1`	3.5778	5.3223	0.3353	8.46
`k4_eps1`	3.5779	5.3223	0.3353	8.38
`muot_dustbin_mono`	3.9030	5.3258	0.4155	8.41
`muot_dustbin_hybrid`	4.6250	5.3244	0.4162	8.45

Table 10: Short TPU v6e-8 Pfam screen across the four core non-dustbin unit-target and dustbin-augmented configurations.

At step 50, the non-dustbin unit-target $R = 2$ and $R = 4$ runs have nearly identical logged loss (3.5778 versus 3.5779) and sparse CE (5.3223 for both to four decimals), with observed throughput 8.46 versus 8.38 examples/s. This is a descriptive short smoke screen rather than an equivalence test or evidence that the runs have fully converged. The single-active-dustbin bridge is operational in the same TPU training harness, with the widened-support cost caveat from Appendix B.1.

C.7 Promoted Non-Dustbin Unit-Target Run

We then promoted the non-dustbin unit-target `k2_eps1` configuration to the longer-budget TPU training path. The full objective details are given in the protocol notes in the appendix; for the main-text interpretation, this is a single-run trainability and checkpoint-persistence test of the factorization-backed $R = 2$ unit-target path on rectangular Pfam pairs. It is neither multi-seed training-stability

evidence nor a compatible balanced-OT experiment when the active sequence lengths differ. The run reached step 1437, repeatedly persisted checkpoints at steps 250, 500, 750, 1000, and 1250, and sustained approximately 8.5 examples per second after startup warmup. The initial throughput at step 0 was 2.67 examples per second, after which the run stabilized near 8.5 examples per second for the remainder of the budget.

Checkpoint / reference	Rec.	Sparse CE	Bary. MAE	Bary. ≤ 5	Mono.
Step 0	5.5730	5.5335	65.1369	0.0406	0.4309
Step 1437	2.0539	5.3046	64.5921	0.0403	0.5003
Diag. ref. (step-0 V)	5.6320	17.8155	1.6367	0.9372	0.0000
Diag. ref. (step-1437 V)	2.0142	17.8155	1.6367	0.9372	0.0000

Table 11: Held-out Pfam test-shard evaluation on 100 examples. "Bary." reports row-normalized transport-barycenter error against supervised target columns. The diagonal references use a length-scaled monotone point-mass alignment; their sparse CE uses a 10^{-12} off-diagonal floor and is not directly comparable to the soft Sinkhorn CE.

Example-bootstrap 95% intervals are $[5.03, 6.06]$ and $[1.78, 2.30]$ for reconstruction at steps 0 and 1437, and $[5.49, 5.58]$ and $[5.26, 5.35]$ for sparse CE. For the step-1437 diagonal reference, the intervals are $[1.73, 2.31]$ for reconstruction, $[1.33, 1.96]$ for barycenter MAE, and $[0.90, 0.97]$ for barycenter within five columns. An untrained same-projection flash-attention reference gives reconstruction 5.5753 and sparse CE 5.5383, matching the step-0 Sinkhorn checkpoint; this remains an initialization sanity reference rather than the trained comparator reported in the preceding six-row screen. The promoted run therefore shows that this factorization-backed path can train for the reported budget, persist checkpoints, and improve reconstruction in one longer run. It does not establish across-seed training stability. Nor does it establish long-horizon across-seed convergence. The CE change is diagnostic; target-barycenter alignment metrics do not materially improve, barycenter-within-five and monotonicity do not improve, and the diagonal reference remains much stronger on alignment metrics. This experiment therefore does not establish an alignment-task advantage.

C.8 Prespecified Alignment-Loss Calibration

To test whether directly optimizing alignment resolves the reconstruction-only limitation, we ran a seed-0 calibration with sparse-CE weights $\lambda_{\text{ce}} \in \{0.1, 0.3, 1.0\}$ against the $\lambda_{\text{ce}} = 0$ baseline. Each row uses the fixed 100-update protocol and the same 5000-example validation evaluator. These are separate short-screen measurements, not extensions of the step-1437 checkpoint or its 100-example test evaluation.

λ_{ce}	Reconstruction MSE	Sparse CE	Barycenter MAE	Within five
0	0.0545948	5.56091	77.3981	0.0435078
0.1	0.0545891	5.55988	77.3928	0.0434887
0.3	0.0545882	5.55984	77.3925	0.0434868
1.0	0.0545910	5.55997	77.3935	0.0434926

Table 12: Seed-0 alignment-objective calibration at the prespecified step 100. Lower is better for the first three metrics; higher is better for within-five accuracy. No tested weight passes the fixed selector.

The selector requires strict improvements in sparse CE, barycenter MAE, and within-five accuracy, while allowing at most 10% relative reconstruction degradation. Every tested weight slightly reduces CE, MAE, and reconstruction MSE, but also reduces within-five accuracy. Consequently no weight or checkpoint is selected, and seed-$1/2$ replication is not triggered. Transient crossings of all four conditions at steps 25 and 75 are post-hoc trajectory diagnostics; they do not change the fixed step-100 decision or justify selecting an intermediate checkpoint. This negative calibration does not establish an alignment advantage or reverse the diagonal-reference result.

C.9 Experimental Protocol Notes

The current manuscript uses eight empirical layers:

1. exactness checks against exact autodiff on small masked problems;
2. a historical T4 dense-reference surface and a same-physical-A100 actual-kernel forward diagnostic against the pinned official FlashSinkhorn package;
3. a literal single-device V5E1 matrix for naive four-resident-plan, fused direct-four, one-reference, and exact-autodiff modes, with separate saved-state backward-only and one-executable complete-step timing, compiler-total memory, and explicit saved state;
4. an all-eight-device v6e-8 explicit complete-VJP follow-up through 262144 positions per device, with historical literal-retention and native softmax context reported separately;
5. a matched three-seed, six-row V5E1-class reconstruction screen for Sinkhorn OT and trained FlashAttention under the same 100-update protocol;
6. a four-configuration TPU v6e-8 topology-transfer Pfam screen across non-dustbin unit-target and dustbin-augmented paths;
7. a promoted long-budget non-dustbin unit-target run for the factorization-backed $R = 2$ path, plus held-out checkpoint and deterministic diagonal-reference evaluation;
8. a seed-0 sparse-CE weight calibration with a fixed step-100 selector and no passing weight.

The topology-transfer Pfam training results use a single Cloud TPU v6e-8 host. The literal systems matrix instead uses one managed TPU V5E1 device; its verified lifecycle lasted 1082.07 seconds, the provenance-bound result bundle was retrieved and atomically imported, and cleanup was verified. The V5E1 publication timings use one fresh top-level process per mode and length, five warmups, and twenty synchronized repetitions. Saved-state backward-only timing and one-compiled-executable complete-step timing are separate contracts. Compiler-total memory is arguments plus outputs plus temporaries minus aliases, with explicit saved payload already included. No independently attributable dynamic XProf peak is available, and no compiler estimate or allocator statistic is substituted for one. The matched six-row screen binds a single-device V5E1 class and topology for every row without claiming the same physical device; its whole-invocation telemetry is not isolated kernel timing. One recovered row is admitted for reconstruction science only and excluded from timing, throughput, systems, and ordinary-success claims. The short v6e-8 screen uses $50{,}000$ Pfam training pairs with the reported benchmark stack, and the promoted run uses the same benchmark family at longer budget. The biological inputs and language-model-derived embeddings are pre-existing assets rather than new data collection.

The extreme-context follow-up in Section C.3 uses a separate v6e-8 allocation and an explicit value-and-VJP interface with VMEM-windowed backward scheduling. Each case has one warmup, three synchronized measurements, and a 350-second walltime limit. The historical literal-retention, dense-Flash, and local-Splash rows instead have two warmups, five measurements, and a 600-second case limit. Exact $Q/K/V/G$ fingerprints, topology, runtime, and configuration fields match where the cross-allocation Sinkhorn comparison is reported, but physical TPU identity and timing protocols do not. The earlier transformation-based low-state routes failed during full-VJP compilation at 131072 and did not attempt 262144; those historical failures are preserved separately from the successful explicit-interface follow-up. The compiler's `pmap` byte scope is reported without assuming per-device or aggregate scaling. Dynamic peak HBM is unavailable.

The alignment calibration in Section C.8 holds the seed-0 short-screen protocol, training inputs, initialization policy, and 5000-example evaluator fixed while adding sparse CE at weights 0.1, 0.3, and 1.0. All decisions use step 100 and the four metric predicates stated there. A passing row would be ranked by lower CE, lower MAE, higher within-five accuracy, and smaller reconstruction degradation, in that order. No row passes, so no selected-weight seed-$1/2$ replication is reported. Intermediate checkpoint evaluations are trajectory diagnostics only.

The historical dense-reference FlashSinkhorn comparison uses one Tesla T4 with 15.64 GB device memory obtained after the requested A100 was unavailable for that allocation. Both frameworks report the host's sole local GPU. The software snapshot is Python $3.12.13$, JAX $0.7.2$, PyTorch $2.11.0{+}\mathrm{cu}128$, and `flash-sinkhorn==0.3.3.post1`; JAX preallocation is disabled. The managed lifecycle lasted 381.48 seconds and cleanup was verified. Table 7 reports synchronized steady-state timings rather than lifecycle time. This run is one seed on one fallback GPU and does not supply uncertainty across devices.

The later actual-kernel diagnostic uses one NVIDIA A100-SXM4-40GB. Every child is bound to the same recorded GPU UUID, and paired rows consume byte-identical $q/k/v$ tensors; this controls physical-device and input identity, not execution-path equivalence. It uses one warmup, five synchronized measurements, float32 storage, dimension 64, and 17 updates. To reach the official dense surface, our TPU-oriented JAX/Pallas program is lowered non-natively through CUDA and its band is expanded to $W = L$, removing the submitted fixed-band regime; the official package instead runs its native Triton CUDA path. Official TF32 is disabled while the archived Pallas path uses JAX's implicit default dot precision; official `exp2` and autotuning have no corresponding Pallas switches, and Pallas output normalization occurs outside timing. These target, regime, arithmetic, and timing caveats preclude an equivalent-native comparison or method-level speed ratio. The managed lifecycle lasted 704.12 seconds, the bundle was retrieved and independently reverified, and cleanup was verified.

For the promoted non-dustbin unit-target run, the objective weights were

$$\lambda_{\mathrm{rec}} = 1, \qquad \lambda_{\mathrm{ce}} = 0, \qquad \lambda_{\mathrm{mono}} = 0,$$

so reconstruction loss was optimized directly while CE and monotonicity were logged diagnostically. Here reconstruction is the mean supervised value-vector MSE: for each active annotated query row i with target column c_i, we compare the transport output $\sum_j P_{ij} V_j$ to V_{c_i}, average squared error over active annotated rows, heads, and value dimensions. The anonymized supplementary code bundle contains the single-file unit-target and dustbin-augmented transport kernel analyzed within the stated theorem boundaries, a Sinkhorn-only Pfam training harness whose defaults match the reported v6e-8 promoted-run settings, the held-out checkpoint evaluator, the exactness-summary JSON used for Table 4, and the held-out metric JSON summaries used for Table 11. It intentionally omits cloud launchers, project-specific storage paths, checkpoints, logs, FlashAttention/FlashSinkhorn training branches, matched-Pfam comparison adapters, the raw managed single-device full-matrix bundle, the full v6e-8 direct-adjoint performance outputs, and external data or embedding generation. The deterministic diagonal reference maps each active query ordinal to the corresponding length-scaled active key ordinal and uses the same value projection as the selected checkpoint. The reported Pfam runs additionally depend on precomputed Pfam/ESM-2 shards and TPU environment setup, so the Pfam claims are scoped to the disclosed protocol notes and reported artifacts rather than to a complete public data pipeline.

For the repository version used to prepare the manuscript, the synthetic diagnostics are reproduced with the JAX environment active by running `python scripts/validate_tail_refinement.py`, `python scripts/validate_orbit_factorization.py`, `python scripts/validate_tail_bias_certificate.py`, and `python scripts/benchmark_four_plan_adjoint.py`. These commands cover the synthetic exactness, orbit, bias-certificate, and local four-plan diagnostics; they do not reproduce the cloud TPU training run without the omitted external shards and environment setup. The supplementary code also includes `benchmark_production_backward_modes.py`, a TPU-facing harness that times the optimized tail-refinement backward and compares gradients against the exact-autodiff reference mode exposed by the same public Sinkhorn API. In the corrected V2 contract, the saved-state backward-only consumer is timed separately from a complete-step executable that contains forward computation, materialization and retention, and arbitrary-cotangent $Q/K/V$ backward. The naive complete-step HLO retains and consumes four distinct plan outputs in that single executable, with marker, barrier, numerical-distinctness, synchronization, and no-dead-code-elimination checks. Exact autodiff has no exposed matched saved-state consumer and is therefore not applicable in the backward-only table, but it remains in correctness and complete-step evidence. Compiler-total memory includes the explicit saved payload; dynamic whole-process XProf peak HBM remains unavailable. This V2 run tests exactly $L = 2048, 8192, 16384$ on one V5E1 device, so it makes no V2 maximum-length claim. A v6e-8 repeat would test topology transfer rather than complete the already measured V5E1 comparison.

The main external assets used in the experiments are public Pfam family data [Mistry et al., 2021] and ESM-2 embeddings from the public ESM-2 checkpoint [Lin et al., 2023]. Pfam is distributed under CC0 through the Pfam/InterPro documentation, and the `esm2_t6_8M_UR50D` weights used for the embeddings are distributed under the MIT license.

The reported dustbin path is the single-active-dustbin augmented unit-target surrogate. KL-unbalanced transport, larger dustbin capacities, sparse-spoke dustbin supports, and differentiable support geometry remain outside the reported scope.

Synthetic diagnostic visual summary. Figure 4 visualizes the exactness and bias-certificate summaries reported in Tables 4 and 5. The left panel is generated from the exactness-summary JSON included in the supplementary bundle. The right panel plots the a posteriori bias-certificate rows from Table 5; the selector tolerance line is the 10^{-5} criterion used to choose $R = 2$ in the local calibration.

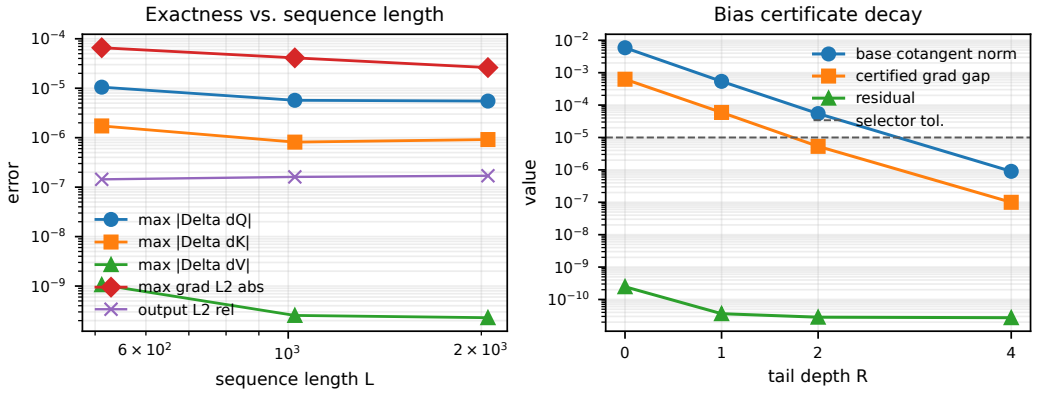

Figure 4: Synthetic tail-refinement diagnostics. Left: optimized $R = 2$ kernel deviations against exact autodiff through the same centered surrogate. Right: a posteriori bias-certificate decay with tail depth.

Qualitative checkpoint output on real Pfam inputs. Figure 5 visualizes a real Pfam/ESM-2 pair generated by the shard pipeline and evaluated with the saved step-0 and step-1437 Sinkhorn checkpoints. The selected pair has 223 query positions, 212 key positions, and 211 supervised alignment targets. For this example, the promoted checkpoint lowers reconstruction MSE from 4.86 to 1.61 and sparse CE from 5.40 to 5.19. The visualization is qualitative: it shows what the saved final checkpoint outputs on real inputs, while the aggregate held-out metrics remain those in Table 11.

Multiple qualitative checkpoint examples. Figure 6 repeats the final-checkpoint transport visualization for four local Pfam/ESM-2 candidates selected by reconstruction-MSE improvement from the first 36 scanned held-out shard examples, subject to at least 180 active targets and maximum visual length 320. The examples have 211–222 supervised targets and improve reconstruction MSE from 4.72–4.86 at step 0 to 1.52–1.61 at step 1437. This figure is a qualitative stress check of the saved checkpoint on multiple real inputs, not an additional benchmark; Table 11 remains the aggregate held-out summary.

Direct four-plan adjoint diagnostic. Table 13 isolates the $R = 2$ score-adjoint stage after the base and tail potentials are already available. The direct baseline forms the four staircase factors $(P^{(2,2)}, P^{(2,1)}, P^{(1,1)}, P^{(1,0)})$ separately; the one-reference implementation forms only $P^{(2,2)}$ and applies the row/column modifiers from Proposition 3. This is a local dense-JAX CPU diagnostic rather than a TPU systems benchmark, so it should be read as a controlled materialization check, not as a FlashSinkhorn comparison or a production wall-clock claim. The fused direct-four comparator in V2 Tables A and B computes four exponentials inside each streaming tile and does not retain four complete plans. The literal naive complete-step mode preserves four complete, distinct plan arrays as outputs of one compiled executable. The CPU diagnostic, fused comparator, and literal resident-plan baseline therefore answer distinct questions.

Analytic memory ledger. Table 14 gives an analytic ledger for the balanced $R = 2$ path at the reported long-context setting $L = 16384$, half-band width $W = 1024$, block size $B = 128$, head dimension $d = 64$, and float32 storage. It compares logical storage inside the $R = 2$ adjoint under explicit plan-factor materialization; it is not a measured comparison against the fused production direct-four implementation, FlashAttention, or FlashSinkhorn workspaces. The command `python scripts/estimate_r2_backward_memory.py` reproduces these values.

The literal V5E1 result verifies the storage distinction directly: its naive complete-step mode retains four float32 plan arrays with 544 MiB total payload at $L = 16384$, while one-reference retains 0.125

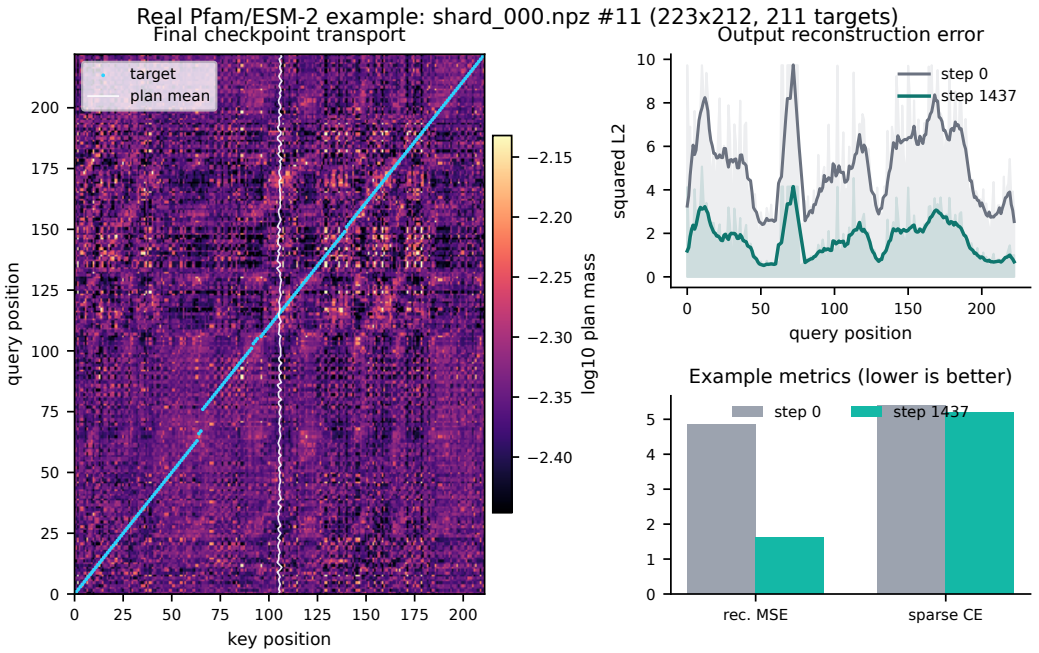

Figure 5: Qualitative model output on a real Pfam/ESM-2 pair. Left: final-checkpoint tail-refinement transport plan, with supervised target columns in cyan and the plan barycenter in white. Right: per-position output reconstruction error and example-level lower-is-better metrics before and after promotion.

L	W	Direct ms	One-ref ms	Speedup	Max $\lvert\Delta\bar{S}\rvert$
512	256	2.32	1.06	2.20×	4.77×10^{-7}
1024	256	7.01	5.44	1.29×	1.19×10^{-7}
2048	256	60.51	39.58	1.53×	2.38×10^{-7}

Table 13: Controlled score-adjoint benchmark for direct four-plan materialization versus the one-reference representation. Logical active plan-factor storage is reduced by exactly $4\times$ in all rows: $3.15/0.79$ MB, $7.35/1.84$ MB, and $15.76/3.94$ MB for direct/one-reference respectively. Timings are mean post-compilation CPU times over 20 repetitions.

MiB of explicit vector state and no full plan array. The corresponding compiler totals are 578.5 and 34.7 MiB, and the saved payload is already included in each total. The present one-reference kernel is nevertheless slower than both direct implementations at the publication lengths. Dynamic XProf peak HBM is unavailable rather than zero, so the evidence supports the exact representation and explicit-residency claim without inferring either a dynamic-HBM or latency advantage from the analytic ledger.

Projective contraction diagnostics on Pfam checkpoints. Table 15 reports the Birkhoff–Hopf coefficient from Proposition 6 on positive score rectangles from the local Pfam test shards for the step-0 and step-1437 Sinkhorn checkpoints. For each selected 128-row block we compute the exact projective-diameter coefficient $\rho_H = \tanh^2(\Delta/4)$ on the scaled scores at $\varepsilon = 1$, excluding inactive query rows and key columns. The diagnostic uses a blockwise column range and does not separately audit each row–column edge against the hard band; therefore these coefficients describe the resulting positive score rectangles and do not certify the full masked solve. We also report the coarser range-only sufficient bound $\rho_{\mathrm{range}} = \tanh^2(\Omega/2)$. Across 228 rectangles from 100 held-out examples, both coefficients remain nontrivial; the final checkpoint has median $\rho_H = 0.241$ and maximum $\rho_H = 0.590$. The increase from step 0 to step 1437 indicates that the trained projections operate at

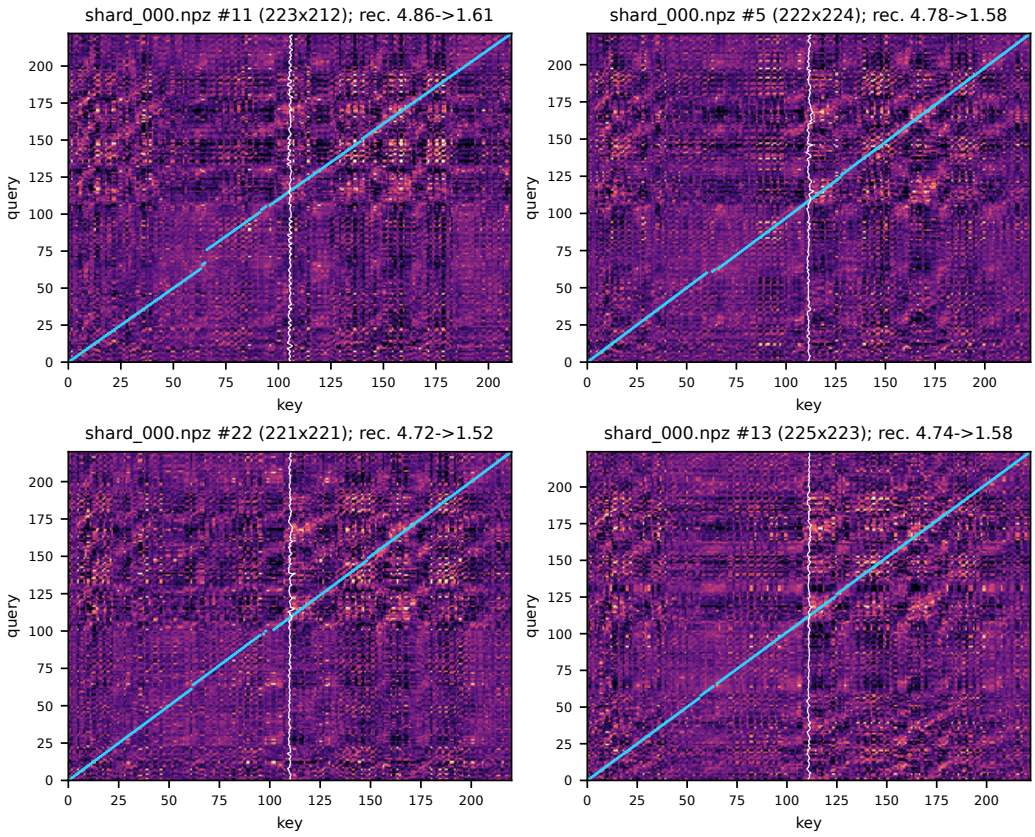

Figure 6: Multiple qualitative Pfam/ESM-2 checkpoint examples selected by local reconstruction-MSE improvement. Each panel shows the final-checkpoint tail-refinement transport plan, supervised target columns in cyan, and the plan barycenter in white. Panel titles report sequence lengths and step-0 to step-1437 reconstruction MSE.

Quantity	Direct four-plan	One-reference
Active support entries per head	32.52M	
Logical materialized plan factors	496.23 MiB	124.06 MiB
Resident plan tile at $B = 128$	0.2500 MiB	0.0625 MiB
$R = 2$ tail u/v vectors	0.375 MiB	
Base-plus-tail u/v history upper ledger	2.25 MiB	
QKV activations	12.00 MiB	

Table 14: Analytic storage ledger for the balanced banded $R = 2$ path at $L = 16384, W = 1024, B = 128, d = 64$ in float32. The first two rows quantify logical plan-factor materialization; the resident-tile row reflects the idealized streaming block schedule. This is not a measured TPU HBM profile or a model of the fused production direct-four comparator.

higher score contrast, while the positive-rectangle maps remain locally contractive in the measured projective metric.

Compact Pfam diagnostic visual summary. Figure 7 visualizes two local empirical summaries requested by the review. The left panel converts the step-50 screen table into a column-normalized heatmap, so the short-screen comparison across loss, sparse CE, monotonicity, and throughput is visible at a glance. The right panel shows the full ρ_H distribution underlying Table 15; it indicates higher learned score contrast, hence weaker but still bounded local contraction margins, rather than improved contraction.

Step	Blocks	Median ρ_H	P95 ρ_H	Max ρ_H	Median ρ_{range}	Max ρ_{range}
0	228	0.0748	0.296	0.345	0.282	0.582
1437	228	0.241	0.532	0.590	0.482	0.721

Table 15: Production-checkpoint projective contraction diagnostics on local Pfam test shards. The exact projective certificate is computed from the active block score matrix; the range-only bound is the simpler sufficient bound stated in Proposition 6.

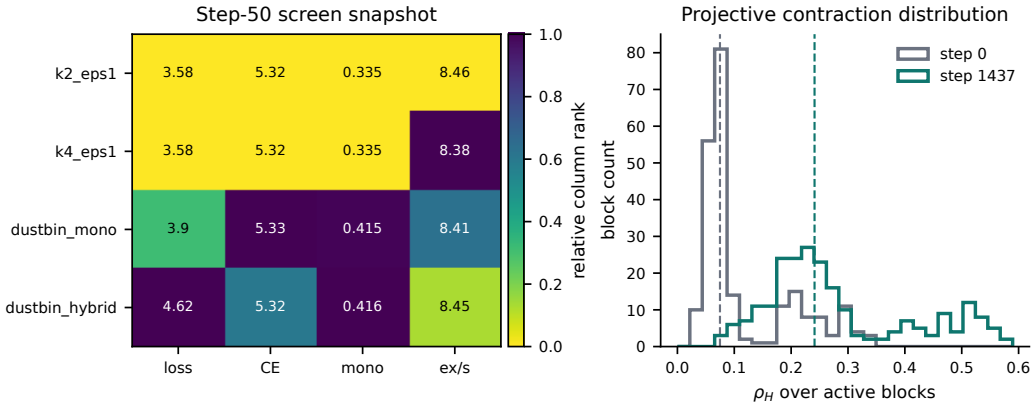

Figure 7: Compact Pfam diagnostic summaries. Left: relative column ranks for the short TPU screen metrics, with annotated raw values. Right: exact projective contraction coefficient distributions over active blocks at the step-0 and step-1437 checkpoints.

D Notation Glossary

The following conventions distinguish objects that play different roles in the proofs.

- $A = QK^\top/\sqrt{d}$ denotes unscaled logits, so $S = A/\varepsilon$. The positive transport kernel is $\mathsf{K} = \exp(S)$ entrywise; K without the sans-serif font continues to denote the key tensor.
- $\bar{S}$ is the cotangent of the scaled score tensor. A bar always denotes a reverse-pass cotangent in the adjoint formulas, whereas a tilde denotes augmented tensors or explicitly centered representatives as specified locally.
- $\mathbf{1}_q$ and $\mathbf{1}_k$ are all-ones vectors on active query rows and key columns. Inactive entries and dustbin fillers are excluded from these vectors and from the active positive cone.
- B denotes tile side length in the streaming analysis, the appended block size in the dustbin construction, and batch size in the V5E1 benchmark shape specification; each occurrence states its local meaning. H is the number of attention heads.
- $\Gamma_c(u, v) = (u - c\mathbf{1}_q, v + c\mathbf{1}_k)$ is a gauge shift. $\mathcal{C}$ chooses a centered representative without changing a same-time plan. Mixed-time plans require the scalar gauge factors recorded in Appendix A.
- x collects differentiable model parameters, $z = (u, v)$ is the dual state, Φ_x is one centered refinement step, and $z_T(x)$ is the base-solve output. The base initial state z_0 is independent of x for the geometric local-bias bound.
- A_x is the vector-valued transport application map in the local-bias analysis; its derivatives J_{full} and J_{surr} are output Jacobians. F_R is the scalar tail objective after the terminal loss, and g_{full} and g_{surr} are its column gradients. η_R is its cotangent on the stopped base state, and $c_R = (D_x z_T)^\top \eta_R$ is the exact omitted-gradient certificate.
- d_H denotes Hilbert's projective metric, $\operatorname{osc}(w) = \max_i w_i - \min_i w_i$, and $\Delta(\mathsf{K})$ is the positive kernel's projective diameter. The projective coefficient ρ_H is a support-local certificate, distinct from an assumed full-state norm contraction constant ρ.

Table 16: Core notation and the objects differentiated in this paper.

Symbol	Meaning
Q, K, V	Query, key, and value tensors; d is their head dimension.
L, W, Ω	Sequence length, band width, and fixed active support.
n_q, n_k	Numbers of active query rows and key columns; these can differ on rectangular Pfam examples.
T, R	Stopped base-solve depth and differentiated refinement-tail depth; the reported factorization uses $R = 2$.
ε	Positive entropic temperature, absorbed into S.
S	Entropic score matrix, $S_{ij} = Q_i K_j^\top / (\sqrt{d}\,\varepsilon)$.
$u^{(t)}, v^{(t)}$	Dual potentials after refinement step t; $(u^{(0)}, v^{(0)})$ is stop-gradient.
$P^{(a,b)}$	Staircase plan with row potential $u^{(a)}$ and column potential $v^{(b)}$ on Ω.
$G, Z, \bar{S}$	Output cotangent, $Z_{ij} = \langle G_i, V_j \rangle$, and the resulting score cotangent. A bar denotes a reverse-pass cotangent.

- $\oslash$ denotes entrywise division and $\odot$ entrywise multiplication, with row/column vectors broadcast over the corresponding matrix index where indicated.

E Kernel-Oriented Pseudocode

Balanced stopped-base and $R = 2$ tail.

1. Form masked scaled scores $S_{ij} = Q_i K_j^\top / (\sqrt{d}\,\varepsilon)$ on the active support Ω.
2. Run the balanced masked Sinkhorn half-steps for T stopped iterations to obtain the base pair $(u^{(0)}, v^{(0)})$.
3. Recompute two differentiable refinement steps from the stopped base pair, producing $(u^{(1)}, v^{(1)})$ and $(u^{(2)}, v^{(2)})$.
4. Materialize only the base staircase plan $P^{(2,2)}$ blockwise.
5. Reconstruct $P^{(2,1)}$, $P^{(1,1)}$, and $P^{(1,0)}$ by row and column rescaling of $P^{(2,2)}$ using the dual differences $(u^{(a)} - u^{(2)}, v^{(b)} - v^{(2)})$. When using stored centered potentials, include the scalar gauge factors from the centered gauge ledger or reconstruct in a gauge-consistent raw representative.
6. Evaluate the exact $R = 2$ score-space VJP from Propositions 2 and 3, then propagate to (Q, K, V).

Dustbin-augmented bridge.

1. Append one active dustbin token per side together with inactive fillers to construct the augmented tensors $(\widetilde{Q}, \widetilde{K}, \widetilde{V})$.
2. Augment the masks analogously and enlarge the effective band width to $\widetilde{W} = \max(W, L)$.
3. Apply the same balanced stopped-base and $R = 2$ tail procedure on the augmented state space.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction match the proved balanced $R = 2$ theorem, the dustbin bridge, the local surrogate-bias bound, the exactness checks, the scoped dense-forward FlashSinkhorn comparison, and the reported TPU runs. They report the literal four-mode single-device V5E1 matrix, including the current negative latency result, explicit resident-plan state, static compiler-accounted totals, and the absence of a dynamic peak-HBM measurement. They also report the completed matched three-seed, 100-update Sinkhorn–FlashAttention reconstruction screen, the explicit exact arbitrary-cotangent VJP at per-device $L = 262{,}144$ on all eight v6e devices, and the negative prespecified seed-0 CE-weight screen. We do not claim a full KL-unbalanced theory, CE-optimized promoted training, a same-allocation four-mode v6e-8 comparison, a matched hard-band FlashSinkhorn Pfam comparison, long-horizon multi-seed convergence, equivalent-hardware speed, or biological-alignment superiority.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The discussion explicitly limits the theorem to fixed-support balanced transport, states that the dustbin bridge is not a full unbalanced OT theorem, and separates logical plan state, static compiler-accounted totals, dynamic peak HBM, scoped VMEM, and latency. It treats the six-row screen as short-horizon descriptive reconstruction evidence and notes that the promoted run optimizes reconstruction while CE is diagnostic only. The separate seed-0 CE screen selects no weight under its prespecified criterion and therefore triggers no seed-$1/2$ replication. The $262{,}144$-token explicit exact-VJP result is a tested point, with separate-allocation comparisons limited to their disclosed protocols; it does not establish a maximum feasible length or universal speed advantage. Long-horizon matched training, stronger alignment evidence, and a same-allocation four-mode topology comparison remain extensions.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The assumptions and statements for the balanced $R = 2$ adjoint, the one-reference-tile backward schedule, the dustbin augmentation bridge, the local surrogate-bias bound, the projective contraction certificate, the a posteriori bias certificate, the certified tail-depth optimization rule, and the transport-orbit calculus are summarized in Sections 3–4 and stated in full, with complete proofs, in Appendices A–B. The local-bias theorem distinguishes output Jacobians from scalar-loss gradients, and the positive-block contraction certificate is explicitly separate from the full-state contraction assumption.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [No]

   Justification: The manuscript discloses the surrogate, kernel schedule, validation settings, TPU types, dataset family, screen and promoted-run settings, checkpoint evaluation protocol, literal V5E1 timing, static compiler-accounted totals, resident-state and feasibility protocols, numerical boundaries, and the official FlashSinkhorn comparison contract. It also describes the explicit exact value-and-VJP route, independent full-sequence execution on each of eight v6e devices, the separate long-context allocation and timing protocols, and the prespecified

CE-weight screen and selector. The supplementary ZIP contains the analyzed kernel, Sinkhorn-only training and evaluation harnesses, evidence JSON, memory-ledger and plotting scripts, and the TPU backward benchmark harness. It omits cloud launchers and storage paths, checkpoints and logs, Pfam/ESM shards and embedding generation, matched-Pfam adapters, trained attention checkpoints, and raw managed results; the cloud experiments, including the long-context follow-up and CE screen, are therefore documented but not fully reproducible from the release bundle.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The anonymized supplement contains the kernel, Sinkhorn-only Pfam training harness, held-out evaluator, evidence summaries, analytic memory ledger, plotting helpers, and the V5E1 production-backward harness. It supports inspection and partial reruns, but not faithful reruns of the completed cloud experiments, trained-attention screen, explicit long-context VJP follow-up, or CE-weight screen because checkpoints, logs, shards, the embedding pipeline, matched-Pfam adapters, raw managed outputs, and cloud launchers are omitted. A matched hard-band FlashSinkhorn experiment and a same-allocation four-mode v6e-8 comparison remain explicitly scoped extensions rather than omitted completed results.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details needed to understand the results?

   Answer: [Yes]

   Justification: The validation and protocol sections specify the masked settings; official FlashSinkhorn package, fallback hardware, timing and correctness contract; literal V5E1 shapes, software, precision, isolated timing, resident-plan, static compiler-accounted totals and feasibility protocols; the VMEM-versus-HBM failure boundary; and the matched six-row methods, seeds, objective, update and evaluation budgets, hardware-class boundary, checkpoint continuity and recovered-row scope. The long-context follow-up uses the explicit exact value-and-VJP route with one independent full sequence per device under eight-device data-parallel `pmap`, without sequence sharding. Its one warmup and three synchronized measurements are distinguished from the separate historical allocation's two warmups and five measurements. The seed-0 CE screen tests weights 0.1, 0.3, and 1.0 at the fixed 100-update endpoint; no weight passes the prespecified all-four-metric selector, so neither an intermediate checkpoint nor seed-$1/2$ replication is selected. TPU screen and promoted-run settings, throughput, and checkpoint milestones are also disclosed. The supplementary kernel and training harness match the analyzed transport path.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The evidence comprises exactness and factorization checks, a three-seed local bias certificate, one same-physical-A100 forced-dense FlashSinkhorn backend diagnostic, one literal four-mode V5E1 matrix, a completed three-seed matched reconstruction screen, a short TPU screen, a single promoted run, an explicit long-context VJP follow-up, and a prespecified seed-0 CE screen. The V5E1 timings use twenty repetitions after five warmups in isolated processes. The all-eight-device long-context follow-up uses three measurements after one warmup, while the separate historical allocation uses five measurements after two warmups; their juxtaposition is descriptive rather than a same-run speed comparison. Static compiler-accounted totals are not dynamic runtime peaks. The matched screen reports every seed, paired differences, sample standard deviation, per-example bootstrap intervals, and a descriptive resampling interval over the three fixed seeds. The CE screen selects no weight and supplies no replicated alignment gain. These experiments do not establish population uncertainty, long-horizon convergence, or uncertainty across physical devices.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The paper identifies the v6e-8 training platform, promoted-run budget and throughput, and masked validation scales. For the literal V5E1 matrix it gives software, shape, precision, the 1082.07-second lifecycle, separate saved-state and complete-step timing protocols, compiler-total memory, the unavailability of an independently attributable dynamic XProf peak, artifact verification, and verified cleanup. For the matched six-row screen it reports the shared single-device V5E1 class/topology and whole-invocation telemetry while stating that physical-device identity was not held constant and one recovered row is science-only. The explicit long-context follow-up reports all-eight-device v6e execution, independent per-device context lengths, timing and compiler-accounted totals, with separate allocations and protocols distinguished. The CE screen reports its seed, weights, fixed 100-update budget, and untriggered replication. For FlashSinkhorn it gives the T4 fallback, memory, software, lifecycle, timing, and cleanup. Hardware-capacity and wall-time details are not reported for every short or local experiment.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics?

   Answer: [Yes]

   Justification: The work is a methods and systems paper on differentiable attention kernels and benchmark training. It does not involve crowdsourcing, human subjects, or prohibited applications, and the manuscript does not make deployment claims beyond the reported experimental scope.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section 6 discusses the potential reduction in memory requirements, the possibility of larger workloads without net energy savings, downstream misuse, and the absence of biological or clinical validation.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The submission does not release a high-risk generative model or a new scraped dataset; it studies transport-based attention kernels and benchmark training methodology.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets used in the paper properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The protocol notes identify the public Pfam data and the public ESM-2 weights used in the experiments and state the license information included in the submission text: CC0 for Pfam/InterPro and MIT for the ESM-2 checkpoint used for embeddings.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

Justification: The submission does not introduce a new public dataset or model asset release.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject experiments.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not part of the core method, proofs, experimental protocol, or scientific claims in this submission.